Hybrid differential evolution algorithms for the optimal camera placement problem

Mathieu Brevilliers, Julien Lepagnot, Lhassane Idoumghar, Maher Rebai, Julien Kritter

Hybrid Differential Evolution Algorithms for the Optimal Camera Placement Problem

Mathieu Brévilliers[(a)], Julien Lepagnot[(a)], Lhassane Idoumghar[(a)], Maher Rebai[(b)] and Julien Kritter[(a)]

[(a)] *IRIMAS (EA 7499) – Université de Haute-Alsace, Mulhouse, France*
[(b)] *Ecole Supérieure d'Ingénieurs Léonard de Vinci, Paris, France*

Abstract:

Purpose – This paper investigates to what extent hybrid differential evolution (DE) algorithms can be successful in solving the optimal camera placement problem.
Design/methodology/approach – This problem is stated as a unicost set covering problem (USCP) and 18 problem instances are defined according to practical operational needs. Three methods are selected from the literature to solve these instances: a CPLEX solver, a greedy algorithm, and a row weighting local search (RWLS). Then, it is proposed to hybridize these algorithms with two DE approaches designed for combinatorial optimization problems. The first one is a set-based approach (DEset) from the literature. The second one is a new similarity-based approach (DEsim) that takes advantage of the geometric characteristics of a camera in order to find better solutions.
Findings – The experimental study highlights that RWLS and DEsim-CPLEX are the best proposed algorithms. Both easily outperform CPLEX, and it turns out that RWLS performs better on one class of problem instances, whereas DEsim-CPLEX performs better on another class, depending on the minimal resolution needed in practice.
Originality/value – Up to now, the efficiency of RWLS and the DEset approach has been investigated only for a few problems. Thus, the first contribution is to apply these methods for the first time in the context of camera placement. Moreover, new hybrid DE algorithms are proposed to solve the optimal camera placement problem when stated as a USCP. The second main contribution is the design of the DEsim approach that uses the distance between camera locations in order to fully benefit from the DE mutation scheme.

Keywords: Optimal camera placement, combinatorial optimization, unicost set covering problem, row weighting local search, differential evolution, hybridization.

Acknowledgments: This work was supported by the French Agence Nationale de la Recherche (ANR) as part of the OPMoPS project (ANR-16-SEBM-0004).

Post-print version of the following published source: Mathieu Brévilliers, Julien Lepagnot, Lhassane Idoumghar, Maher Rebai, Julien Kritter, (2018) "Hybrid differential evolution algorithms for the optimal camera placement problem", Journal of Systems and Information Technology, Vol. 20 Issue: 4, pp.446-467, https://doi.org/10.1108/JSIT-09-2017-0081

1 Introduction

Nowadays, camera networks are widely used to monitor areas of interest. When connected to an intelligent video surveillance system, it can help to automatically identify targets, events or risks, depending on the given operational requirements. In this context, determining the optimal placement of the cameras is of high importance because of the underlying costs.

For this kind of optimization problem, the area to be monitored and the camera parameters (space coordinates and orientation angles) can be discretized in order to define the following decision problem: given a set of candidate camera locations that cover some discrete points of the area, find an optimal subset that satisfies the operational constraints (Horster and Lienhart, 2009).

Up to now, several coverage models have been proposed in the literature (Mavrinac and Chen, 2013; Zhang et al., 2015). In this paper, the problem is stated as a unicost set covering problem (USCP), and this USCP model is used together with a three-dimensional model of the monitored area. As already noticed in the literature, this 3D setting allows to avoid blind spot due to major simplifications in a 2D setting (Zhang et al., 2013), but it leads to a significant increase of the computational cost (Liu et al. 2016). For example, a bi-objective variant of the problem (minimizing the total cost of the camera network, while maximizing the area coverage) has been recently solved optimally with exact methods (Rebai et al., 2016), but at least 4 hours of computation were needed for the largest instance, which was limited to a 3D grid of 15x15x7 discrete 3D points. It is thus of high interest to design new algorithms that can find high quality solutions in this much larger 3D search space. In this work, the full coverage constraint is also considered: on the one hand, no blind spot is allowed (which can be a strict requirement for some applications) and on the other hand, it is known to make easier the development of person tracking algorithms (Liu et al. 2016).

According to a recent comprehensive survey (Liu et al. 2016), a wide range of methods have already been implemented to solve different variants in this class of problems. Actually, the optimal camera placement problem is often tackled by using binary integer programming methods at first (David et al. 2007; Horster and Lienhart, 2009). However, as soon as the size of the problem increases, these methods can not find an optimal solution within a reasonable run time. That's why approximation methods were also designed, including greedy heuristics (Horster and Lienhart, 2009; Zhao, 2011), semi-definite programming (Ercan et al., 2006; Zhao, 2011), simulated annealing algorithms (Zhao, 2011; Liu et al., 2014), genetic algorithms (David et al. 2007; Van den Hengel et al., 2009), particle swarm optimization algorithms (Morsly et al., 2012; Konda and Conci, 2013), and artificial bee colony algorithms (Chrysostomou and Gasteratos, 2012).

This article focuses on a metaheuristic called differential evolution (DE), which was originally designed for solving continuous optimization problems (Storn and Price, 1997). This simple and efficient evolutionary algorithm is able to solve various theoretical and real-world optimization problems (Das et al., 2016). In DE, a population of individuals (i.e. candidate solutions) is evolving from generation to generation in order to converge on the global best solution. A generation is composed of three evolutionary operators. Firstly, a mutation operator creates a mutant individual by adding weighted differences to a reference individual. The most common DE mutation scheme, called DE/rand/1, is formulated as follows. For each variable j of each individual i of the population Pop:

$$\mathrm{Mut_{i,j}} = \mathrm{Pop_{r_1,j}} + \mathrm{F} \times \left(\mathrm{Pop_{r_2,j}} - \mathrm{Pop_{r_3,j}}\right) \quad (1)$$

where Mut refers to the mutant population, r_1 , r_2 and r_3 to three randomly chosen individuals of Pop such that $r_1 \neq r_2 \neq r_3 \neq i$, and $F\epsilon[0,1]$ to the DE scaling factor. As soon as a variable gets out of the search space due to Equation 1, a new appropriate random value is generated. Secondly, a crossover operator is applied, generating a trial individual from a current individual and its corresponding mutant individual by using the classical so-called binomial crossover. Thirdly, a selection operator replaces any current individual in Pop with its corresponding trial individual, if the latter performs better.

To the best of our knowledge, only one study applied a DE algorithm in order to optimize the camera placement (Zhang et al., 2016). However, the considered problem significantly differs from the one formulated above. Indeed, the area to be covered is represented by the set of triangles in an input 3D triangular mesh, the number of camera to be placed is known in advance, and the aim is to maximize the number of covered triangles. Moreover, the classical above-mentioned DE algorithm is implemented, and it is only compared to a greedy approach.

This paper proposes to investigate the efficiency of two hybrid DE approaches in order to solve the optimal camera placement problem. The first one is a set-based method designed to solve general combinatorial optimization problems (Maravilha et al., 2013). DE has already been adapted to combinatorial optimization in various ways, but most of these adaptations can only be applied on permutation-based combinatorial optimization problems, and even the more general list-of-movements approach (Prado et al., 2010) is not well-suited for tackling the optimal camera placement problem considered here. So, the set-based DE approach seems to be the most appropriate method from the literature. Moreover, it provides promising results when solving the capacitated centered clustering problem and the traveling salesman problem (Maravilha et al., 2013; Maravilha et al., 2014). The second one is a new similarity-based method that takes advantage of the geometric components of a camera location (i.e. space coordinates and orientation angles): it allows to make sense of the DE mutation scheme in this camera placement application.

According to Talbi's taxonomy of hybrid metaheuristics (Talbi, 2002), any implementation of both approaches is a low-level teamwork hybrid (LTH) algorithm. Actually, the mutation operator allows to define a much smaller subproblem, and the crossover consists in solving this subproblem with any appropriate method. Three state-of-the art algorithms have been selected for hybridization with these DE approaches: a CPLEX optimizer (IBM, 2017a), a greedy algorithm (Johnson, 1974), and a row weighting local search (RWLS) algorithm (Gao et al., 2015). CPLEX and the greedy algorithm are natural candidates to get first benchmark results: the former highlights where the limit of an exact method is, while the latter provides a first upper bound for instances that are beyond this limit. Furthermore, RWLS has been experimentally shown to be one of the best heuristic algorithms when solving a large set of USCP benchmark problems. Regarding practical applications, RWLS has been already implemented to solve test suite reduction problems (Chi et al., 2017). But, up to now, no use of RWLS has been reported for solving the optimal camera placement problem.

The remaining of this article is organized as follows. Section 2 describes in detail the considered optimal camera placement problem, specifies the problem modelling, and defines a set of instances inspired by real-world applications. Section 3 first presents CPLEX, the greedy algorithm, RWLS, the set-based DE approach, and the new similarity-based DE approach. Then, it explains the

experimental settings, and the reported results are discussed. Finally, Section 4 sums up the contribution of this article and gives some perspectives for future work.

2 Problem description

2.1 Problem modelling

This paper deals with the following optimal camera placement problem: 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 monitored area is a rectangular box whose point coordinates 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, where X_{max}, Y_{max} and Z_{max} are user-defined values. This area is discretized and approximated by a regular grid of points, where the step size U between two adjacent points is a user-defined parameter. If each of these points is covered by at least one camera, the area is said to be fully covered by the cameras.

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:

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

Any point of the monitored area is said to be covered by a camera if it lies in the pyramid of vision of this camera.

A camera location is characterized by a point in the considered discrete grid together with discrete pan and tilt angles. Camera coordinates can range from $(0,0,Z_{min}^{cam})$ to $(X_{max}, Y_{max}, Z_{max}^{cam})$ with a step size U, where Z_{min}^{cam} and Z_{max}^{cam} are user-defined values such that $Z_{max} \leq Z_{min}^{cam} \leq Z_{max}^{cam}$. A camera can thus be placed anywhere in the grid, provided that it is above or at least on the top of the monitored area. The angular orientation of a camera is then given by two angles: α is the pan angle, that is the rotation angle of the camera 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 user-defined integer A, which fixes the step size to the value π/A. It means that α can take N_α=2A different values that range in $[0,2\pi[$. Regarding β, one can see that values in $]\pi, 2\pi[$ are not needed, since each camera is placed above the points to be covered, and thus, it has to be oriented downward. Moreover, since α ranges in $[0,2\pi[$, any camera location with pan angle α and tilt angle $\beta = k\frac{\pi}{A}$ such that $\beta < \frac{\pi}{2}$, will be identical to the camera location with same coordinates and pan

angle $\alpha' = \alpha + \pi$ and tilt angle $\beta' = \pi - k\frac{\pi}{A}$. It means that β can be limited to $N_\beta = \lfloor A/2 \rfloor + 1$ different values that range in $\left[0, \lfloor A/2 \rfloor \times \frac{\pi}{A}\right]$.

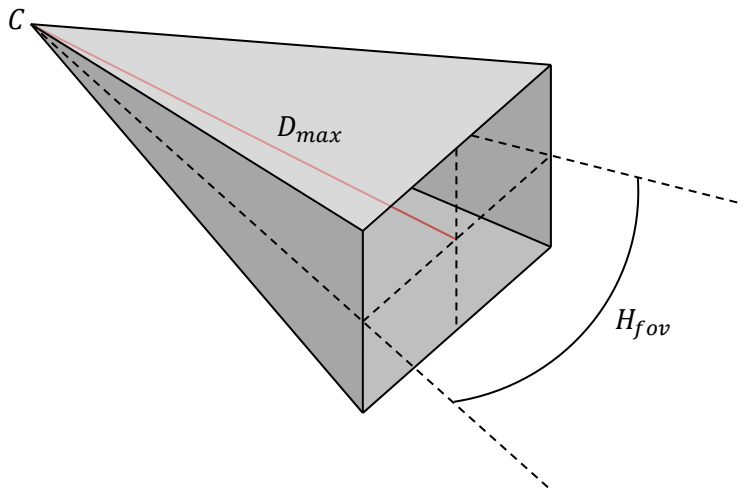

Figure 1: Example of a camera C with horizontal field of view H_{fov}, and whose pyramid of vision has height D_{max}.

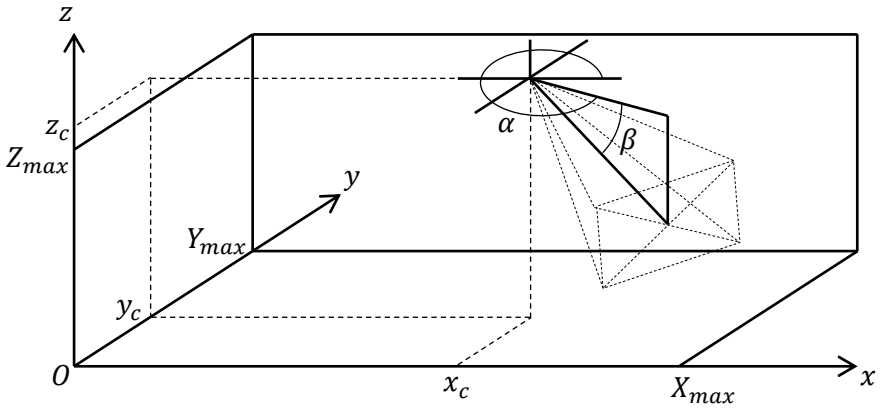

Figure 2: Example of camera location with coordinates (x_c, y_c, z_c), pan angle α, and tilt angle β.

Given that the objective is to find as few cameras as possible that can completely cover the space to be monitored, it follows that this camera placement problem can be formulated as a unicost set covering problem (USCP) in a straightforward manner. Actually, the points of the monitored area can be labelled with integers, representing the set of elements to be covered. Each camera location can then be modelled as a set of integers, corresponding to the labels of the points it covers. Now, given the set E of elements (i.e. points) and a collection S of sets (i.e. camera locations), solving the optimal camera placement problem comes down to find the minimum subset of S that covers E.

Once the problem is stated as a USCP, the following decision variables can be defined:

$$\forall c \epsilon S, x_c = \begin{cases} 1 & \text{if camera location } c \text{ is used,} \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

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

$$Min \sum_{c \epsilon S} x_c \tag{4}$$

subject to

$$\forall p \epsilon E, \sum_{c \epsilon S: p \epsilon c} x_c \geq 1 \tag{5}$$

$$\forall c \epsilon S, x_c \epsilon \{0,1\}. \tag{6}$$

The objective function (see Equation 4) minimizes the total number of used cameras. The set of constraints (see Equation 5) indicates that each point of E has to be covered by at least one camera location of S: it ensures the full coverage of the monitored area. Equation 6 gives the set of binary constraints needed for the decision variables (see Equation 3).

2.2 Problem instances

In this study, the efficiency of the proposed methods is investigated by using several problem instances inspired by real-world settings (see Table 1). In these instances, the size of the monitored area goes from 5×5×2 meters to 70×70×2 meters. The areas are discretized with a step size of 0.5 meter. Cameras have a resolution of 1920×1080 pixels with a horizontal field of view of 65 degrees, and are supposed to be fixed to the ceiling at a standard height of 2.5 meters ($Z_{min}^{cam} = Z_{max}^{cam}$). The pan and tilt angles are discretized with a step size of $\frac{\pi}{4}$.

Table 1: List of instances.

Instance	X_{max}	Y_{max}	Z_{max}	Z_{min}^{cam}	Z_{max}^{cam}	U	$OpNeed$	H_{res}	V_{res}	H_{fov}	A
1	5	5	2	2.5	2.5	0.5	100	1920	1080	65	4
2	10	10	2	2.5	2.5	0.5	100	1920	1080	65	4
3	15	15	2	2.5	2.5	0.5	100	1920	1080	65	4
4	20	20	2	2.5	2.5	0.5	100	1920	1080	65	4
5	25	25	2	2.5	2.5	0.5	100	1920	1080	65	4
6	30	30	2	2.5	2.5	0.5	100	1920	1080	65	4
7	40	40	2	2.5	2.5	0.5	100	1920	1080	65	4
8	50	50	2	2.5	2.5	0.5	100	1920	1080	65	4
9	5	5	2	2.5	2.5	0.5	500	1920	1080	65	4
10	10	10	2	2.5	2.5	0.5	500	1920	1080	65	4
11	15	15	2	2.5	2.5	0.5	500	1920	1080	65	4
12	20	20	2	2.5	2.5	0.5	500	1920	1080	65	4
13	25	25	2	2.5	2.5	0.5	500	1920	1080	65	4
14	30	30	2	2.5	2.5	0.5	500	1920	1080	65	4
15	40	40	2	2.5	2.5	0.5	500	1920	1080	65	4
16	50	50	2	2.5	2.5	0.5	500	1920	1080	65	4
17	60	60	2	2.5	2.5	0.5	500	1920	1080	65	4
18	70	70	2	2.5	2.5	0.5	500	1920	1080	65	4

There are two main classes of instances (1 to 8, and 9 to 18) that differ in the operational need: 100 or 500 pixels per meter. The aim is to provide adequate resolutions for face recognition, human detection and gait recognition applications, as discussed below.

Regarding automatic face recognition, current commonly-used methods, such as principal component analysis (PCA), linear discriminant analysis (LDA) and local binary pattern (LBP), can reach high success rates (from 70% to 100%) with face image resolution of at least 64×64 pixels (Huang and Wang, 2008; Marciniak et al., 2015; Mahmood et al., 2016). In addition to that, according to recent anthropometric studies (Zhuang et al., 2010; Gordon et al., 2014), the average face width of human people is about 14 centimeters, and the average head height is about 23.5 centimeters. Now, when considering an operational need of 500 pixels per meter, any face covered by a camera will have a resolution of at least 70×118 pixels, which meets the above-mentioned face recognition requirements. It also satisfies the requirements from the European norm "EN 50132-7: CCTV and alarm systems" (Marciniak et al., 2015), where a resolution of at least 330 pixels per meter is recommended for precise identification.

For automatic human detection, it has been shown that a high success rate (higher than 70%) can be achieved with a resolution between 20 and 60 pixels per meter (Miyazaki et al., 2015). In the field of gait recognition, a recent study shows that current methods perform very well (success rate higher than 90%) with a resolution of at most 140 pixels per meter (Liang et al., 2016). However, another work points out that good results can also be achieved with lower resolution between 10 and 80 pixels per meter (Zhang et al., 2010). In this paper, a resolution of 100 pixels per meter is considered as the requirement for automatic human detection or gait recognition applications.

It is also worth noting that the selected operational needs are consistent with the requirements suggested by network video companies (Axis, 2017).

2.3 Data pre-processing

In order to solve the problem instances given in Section 2.2, they have to be processed in order to become standard USCP instances as defined in Section 2.1. Two types of pre-processing are implemented. The first one consists in computing the coverage of each possible camera location. The second one aims at reducing the problem by removing useless camera locations.

For any instance whose characteristics are given in Table 1, the set of points and the set of possible camera locations can be created. Then, for each camera location and for each point, it has to be decided whether this point is visible or not from this camera location: the resulting sets of covered points correspond to the input sets needed for the USCP. The visibility test is performed in the following way (Zhang et al., 2013): new coordinates of the tested point are computed in a coordinate system centered on the camera, i.e. the origin is the camera position, the pyramid height from the base to the apex is included in the X axis, and the length of the rectangular base is parallel to the Y axis (see Figure 3).

So, the original coordinates $P = [x, y, z]$ are transformed into the new ones $P' = [x', y', z']$ with the help of homogeneous coordinate transformations by using the following equation:

$$P' = PTR_Z R_Y \tag{7}$$

Where $T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_c & -y_c & -z_c & 1 \end{bmatrix}$, $R_Z = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, $R_Y = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ are respectively: a translation such that $[x_c, y_c, z_c]$ are the camera coordinates in the monitored

area, a rotation about the Z axis by angle α, and a rotation about the Y axis by angle β. Once P' is computed, the tested point lies inside the pyramid of vision if the following conditions are met:

$$0 \leq x' \leq D_{max} \tag{8}$$

$$|y'| \leq \frac{1}{2} \times \frac{H_{res}}{OpNeed} \times \frac{x'}{D_{max}} \tag{9}$$

$$|z'| \leq \frac{1}{2} \times \frac{V_{res}}{OpNeed} \times \frac{x'}{D_{max}} \tag{10}$$

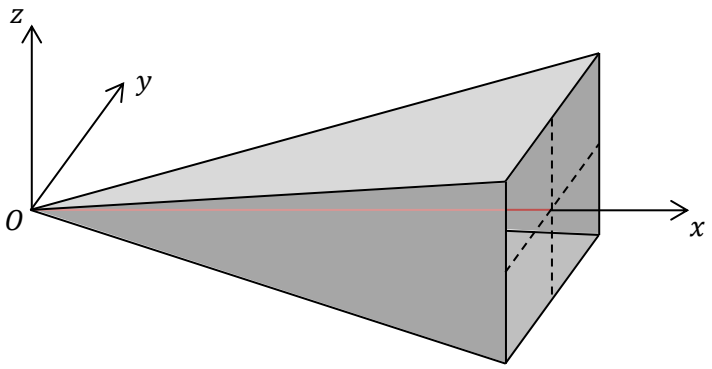

Figure 3: Coordinate system centered on the camera.

It is worth noting that these geometric computations are costly when increasing the size of the instance: for example, 1 756 920 visibility tests are needed for instance 1 (605 points and 2 904 camera locations), and 1 661 500 920 visibility tests are needed for instance 6 (18 605 points and 89 304 camera locations). This remark is even more important if A is increased to get better angles, if U is increased to get a better coverage, or if the cameras are allowed to be placed at different heights (i.e. $Z_{min}^{cam} \neq Z_{max}^{cam}$). However, since the visibility tests are independent, this first type of pre-processing can benefit from the SIMD architecture of GPU devices in order to accelerate the geometric computations. Indeed, compared to the sequential C/C++ implementation on a Intel Core processor i5-3330 CPU (3.00GHz) with 4 GB of RAM, up to a 15 times speedup can be observed by using a simple CUDA C implementation on a NVIDIA GeForce GTX680. A better acceleration can certainly be achieved with a more clever and fine-tuned GPU implementation and with a more powerful GPU device.

The second type of pre-processing consists in reducing the resulting USCP instance, mainly by decreasing the number of possible camera locations. At first, the camera locations that can not cover any point in the monitored area are removed. Then, so called dominated camera locations are removed: a camera location c is said to be dominated by another camera location d if d covers at least the same points as c.

These two types of pre-processing allow to provide reduced USCP instances related to the original optimal camera placement problems given in Table 1. Generally, the USCP input data are presented as a zero-one matrix, where the rows are the elements, and the columns are the sets: a one in row i and column j means that the i-th element is covered by the j-th set. According to this remark, the characteristics of the resulting USCP instances are presented in Table 2: instance number, number of rows (i.e. elements to be covered, or points to be monitored), number of columns (i.e. sets of the

USCP, or available camera locations) in the reduced instance and in the original instance in brackets, percentage of ones and maximum number of ones per row in the corresponding sparse matrix. The corresponding input data files are available on line for download[1].

Table 2: Characteristics of the reduced USCP instances.

Instance	Rows (i.e. elements, or points)	Columns (i.e. sets, or camera locations)	Density of ones (%)	Maximum number of ones per row
1	605	1 292 (2 904)	12.9	292
2	2 205	908 (10 584)	13.7	628
3	4 805	924 (23 064)	38.8	684
4	8 405	4 572 (40 344)	24.9	2 144
5	13 005	9 852 (62 424)	17.0	3 484
6	18 605	16 732 (89 304)	12.3	4 568
7	32 805	35 291 (157 464)	7.2	4 656
8	51 005	60 251 (244 824)	4.8	4 656
9	605	1 672 (2 904)	6.8	212
10	2 205	7 352 (10 584)	2.0	216
11	4 805	17 032 (23 064)	0.9	216
12	8 405	30 712 (40 344)	0.5	216
13	13 005	48 392 (62 424)	0.4	216
14	18 605	70 072 (89 304)	0.2	216
15	32 805	125 431 (157 464)	1.4e-3	216
16	51 005	193 791 (244 824)	9.1e-4	216
17	73 205	284 151 (351 384)	6.3e-4	216
18	99 405	387 511 (477 144)	4.7e-4	216

3 Optimization methods

3.1 State-of-the-art algorithms

This section presents the three state-of-the art algorithms selected for solving the optimal camera placement problem defined in Section 2. These algorithms will also be used in the next sections in order to design new hybrid algorithms and to see to what extent these hybridizations can help to improve the solution found so far.

The first algorithm is IBM ILOG CPLEX optimizer (IBM, 2017a), which is commonly used for solving large integer programming problems, including the USCP. As noticed in the literature (Yelbay et al., 2015; Demirović et al., 2016), general purpose optimizers can solve quite easily weighted instances of the set covering problem, but USCP instances are much harder to solve. In this paper, it is interesting to investigate where is the limit of CPLEX for the practical application of USCP in the context of camera placement, and to compare it with some state-of-the-art and also some new approximation algorithms.

The second algorithm is a greedy one that uses a very intuitive idea to solve the USCP: starting from an empty solution, iteratively add in the solution the set that maximizes the number of new covered elements, i.e. elements covered by this set that were not covered so far (Johnson, 1974). Table 3 shows how this idea is adapted to the context of optimal camera placement. With an adequate implementation (i.e. with $O(n \log n)$ complexity), such a greedy algorithm can quickly provide a

feasible solution. But, it has been proven that the size of a solution given by this algorithm is at most H_d times the size of an optimal solution (Johnson, 1974; Chvatal, 1979), where $H_i = \sum_{j=1}^{i} \frac{1}{j}$ is the i-th harmonic number and d is the size of the largest set $cov(c), \forall c \in C$, knowing that C is the set of possible camera locations. In this paper, the greedy algorithm is used as a benchmark when CPLEX fails solving the instances defined in Section 2.2.

Table 3: Greedy algorithm for the optimal camera placement problem formulated as a USCP.

	Input :	The set C of possible camera locations. The set P of points to be covered. $\forall c \in C, cov(c) = \{p \in P: c \text{ covers } p\}$.
	Output :	A set S of camera locations that covers P, i.e. such that $\bigcup_{c \in S} cov(c) = P$.
1	$S = \emptyset$	
2	**While** $\bigcup_{c \in S} cov(c) \neq P$ **do**	
3	$S = S \cup \{b\}$, where $b \in C \backslash S$ such that $\lvert cov(b) \rvert$ is maximized	
4	$\forall c \in C \backslash S, cov(c) = cov(c) \backslash cov(b)$	
5	**End while**	

The third algorithm is a row weighting local search (RWLS) algorithm (Gao et al., 2015). The main feature of RWLS is its row weighting scheme that helps to identify hard-to-cover rows and to prioritize the columns to be selected in the candidate solution. Each row starts with a weight of 1. Then, after each iteration of the local search procedure, the weights of the uncovered rows are increased by 1. Since the local search first removes sets to get a partial solution and, then, add a new set to try to get a full coverage, the set of uncovered rows is changing at each iteration. Thus, with time, the rows that are harder to cover will get larger weights, given that they will be more often uncovered. These weights are used to define a score for each column as follows. On the one hand, when a column is not in the candidate solution, its score is set to the sum of the weights of all uncovered rows it can cover. Thus, the more a column can cover uncovered rows, especially hard-to-cover uncovered rows (i.e. with larger weights), the more this column has a higher score, and the more it is likely this column will be selected to be added in the candidate solution. On the other hand, when a column is part of the candidate solution, its score is set to the negation of the sum of the weights of rows which are only covered by this column in the candidate solution. Thus, the more a column of the candidate solution is the only one that covers some rows, especially hard-to-cover rows (i.e. with larger weights), the more this column has a lower score, and the less it is likely this column will be removed from the candidate solution. The outline of RWLS is given in Table 4.

This row weighting scheme together with two tabu strategies and a timestamp method allow RWLS to be very efficient (Gao et al., 2015). Actually, to the best of our knowledge, RWLS is the heuristic algorithm that leads to the best results when solving a large number of instances from the OR-Library (Beasley, 1990) and Steiner triple systems (Fulkerson et al., 1974). In the case of optimal camera placement, RWLS is used as an other (and more interesting) benchmark in order to highlight the benefits or the loss of the methods proposed in Sections 3.2 and 3.3.

3.2 Set-based DE approach

This section presents a set-based DE approach that was designed to solve general combinatorial optimization problems (Maravilha et al., 2013). In their work, a solution is formulated as a subset

(instead of a permutation) of combinatorial elements: for instance, a solution of the travelling salesman problem (TSP) is a subset of all possible edges between cities, instead of a permutation of all cities to be visited. In the case of the optimal camera placement problem, a solution is a subset of all the possible camera locations.

Table 4: Outline of RWLS for the optimal camera placement problem.

	Input:	The set C of possible camera locations. The set P of points to be covered. $\forall c \in C, cov(c) = \{p \in P: c \text{ covers } p\}$.
	Output:	A set S_{best} of camera locations that covers P, i.e. such that $\bigcup_{c \in S_{best}} cov(c) = P$.
1	Greedily compute an initial solution S	
2	Initialize point weights and camera location scores according to S	
3	**While** the stopping condition is not met **do**	
4	**While** $\bigcup_{c \in S} cov(c) = P$ **do**	
5	Update S_{best} with S if $\lvert S \rvert < \lvert S_{best} \rvert$	
6	Remove from S the camera location with the highest score	
7	**End while**	
8	Remove from S the camera location with the highest score	
9	Randomly select an uncovered point p	
10	Add in S the camera location with the highest score and that covers p	
11	Update point weights and camera location scores	
12	**End while**	

According to this representation of the solution, DE/rand/1 mutation scheme is modified by using operations on sets in the following way, for each individual i in the current population Pop:

$$Mut_i = Sol_{rand} \cup F \cdot \left(Pop_{r_1} \oplus Pop_{r_2}\right) \tag{11}$$

Where Sol_{rand} is a randomly generated feasible solution, $Pop_{r_1} \neq Pop_{r_2} \neq Pop_i$ are individuals randomly chosen in the current population, and $\oplus$ is the XOR operator on sets. The arithmetic operations of the original DE/rand/1 mutation operator (see Equation 1) are replaced by union and XOR operations: given that the individuals are sets, these operations can be applied in a straightforward manner. Regarding the scaling factor F, the authors suggest to use one of the strategies defined in the literature (Prado et al., 2010) in order to control the size of the resulting set Mut_i. Actually, in this paper, neither of those strategies is preferred since F is set to 1 for the experimental study (see Table 9), which means that no elements are removed from the sets.

Then, the crossover operator generates a trial solution by selecting only elements that are present in $Pop_i \cup Mut_i$. In other words, creating a good trial solution comes down to solve a subproblem of the original one. Since this subproblem is much smaller, the authors suggest to solve it with exact algorithms.

It is proposed here to hybridize this set-based DE approach with the three state-of-the-art algorithms given in Section 3.1, in order to solve the optimal camera placement problems defined in Section 2. Table 5 gives the outline of the set-based DE approach (DEset, for short) in this context.

3.3 Similarity-based DE approach

In the original DE/rand/1 mutation scheme (see Equation 1), the equation is applied for each variable j of each individual i. It makes sense when solving continuous optimization problems, where each decision variable represents one characteristic of the problem. However, in the case of the optimal camera placement problem, a solution is a set of unordered camera locations: it is the same solution whatever the permutation of its camera locations. Moreover, solutions in the population can have different sizes, i.e. two solutions can have different numbers of camera locations. Thus, it has no real sense to directly apply such a mutation equation in this context.

Table 5: Outline of DEset for the optimal camera placement problem.

	Input:	The set C of possible camera locations. The set P of points to be covered. $\forall c \in C, cov(c) = \{p \in P: c \text{ covers } p\}$.
	Output:	A set S_{best} of camera locations that covers P, i.e. such that $\bigcup_{c \in S_{best}} cov(c) = P$.
1	Generate a population Pop of random feasible solutions	
2	Initialize S_{best} with the best solution of Pop	
3	**While** the stopping condition is not met **do**	
4	**For** each individual Pop_i **do**	
5	Randomly select Pop_{r_1} and Pop_{r_2} such that $r_1 \neq r_2 \neq i$	
6	Generate Mut_i by following Equation 11	
7	Generate a trial solution T_i by solving the optimal camera placement subproblem where $C = Pop_i \cup Mut_i$	
8	Update Pop_i and eventually S_{best} if T_i is better	
9	**End for**	
10	**End while**	

The main feature of the similarity-based DE approach aims at overcoming this drawback by improving the DE mutation operator. Here, it generates a mutant individual Mut_i for each individual i of Pop, by using the following DE/rand/1-like mutation equation:

$$Mut_{i,j} = Pop_{r_1,j} + F \times \left(Pop_{r_2,u_j} - Pop_{r_3,v_j}\right), \tag{12}$$

Where $Pop_{r_1} \neq Pop_{r_2} \neq Pop_{r_3} \neq Pop_i$ are individuals randomly chosen in the current population, and u_j and v_j are camera locations of Pop_{r_2} and Pop_{r_3} selected according to the similarity rule explained hereafter. The key point is that the camera locations coming from Pop_{r_2} and Pop_{r_3} are selected depending on their similarity with the camera location coming from Pop_{r_1}. Actually, for each camera location j of Pop_{r_1}, the camera location u_j of Pop_{r_2} that is the most similar to $Pop_{r_1,j}$ is selected. In the same way, the camera location v_j of Pop_{r_3} that is the most similar to $Pop_{r_1,j}$ is selected. This strategy is inspired by the similar-metavariable recombination for genetic algorithms, in order to solve variable-length optimization problems (Ryerkerk et al., 2017). But, it is applied here in the mutation operator and with a different similarity definition. Actually, the similarity is defined as the Euclidean distance as follows:

$$d_{c_1,c_2} = \sqrt{\left(x_{c_2} - x_{c_1}\right)^2 + \left(y_{c_2} - y_{c_1}\right)^2 + \left(z_{c_2} - z_{c_1}\right)^2 + \left(\alpha_{c_2} - \alpha_{c_1}\right)^2 + \left(\beta_{c_2} - \beta_{c_1}\right)^2} \tag{13}$$

Where c_1 and c_2 are two camera locations whose coordinates are $(x_{c_1}, y_{c_1}, z_{c_1})$ and $(x_{c_2}, y_{c_2}, z_{c_2})$ respectively, and whose orientation angles are $(\alpha_{c_1}, \beta_{c_1})$ and $(\alpha_{c_2}, \beta_{c_2})$ respectively.

A first remark is that the size of Mut_i is the same as Pop_{r_1}, i.e. Mut_i contains exactly as many camera locations as Pop_{r_1}. It is also worth noting that a camera location c can be defined as a metavariable which is composed of five design variables $(x_c, y_c, z_c, \alpha_c, \beta_c)$. Thus, Equation 12 is actually applied on each of these design variables, and the resulting real values are rounded down in order to generate a camera location that exists in the discrete search space. Due to the arithmetic computations in Equation 12, the design variables of Mut_i can take values that are out of the search space. In case of a coordinate, it is randomly regenerated inside the search space. In case of an angle, possible values are considered as a cycle from which a new feasible value is deduced. Then, it remains to check whether the camera locations of Mut_i are desirable camera locations or not. Firstly, each dominated camera location is replaced with the one which dominates it. Secondly, redundant or blind camera locations are removed.

The second main feature of the approach proposed here is a diversification strategy that helps the algorithm to escape from local optima. Stagnation is detected by using a user-defined integer g that defines how many generations are allowed without improving the best solution found so far. In case of stagnation, each individual of the current population is replaced by a randomly generated feasible solution with a user-defined probability ρ.

The crossover operator used here in the DE framework is the same as the one defined in Section 3.2: it has to solve the optimal camera placement subproblem with camera locations in $Pop_i \cup Mut_i$. Given that Mut_i has the same size as Pop_{r_1}, it follows that these subproblems are smaller than those of DEset. In the same way as DEset, this similarity-based DE approach (DEsim, for short) is hybridized with the algorithms given in Section 3.1. Table 6 gives the outline of the DEsim approach in the context of optimal camera placement.

3.4 Experimentations

In this section, two experimentations are presented. The first one aims at comparing the three state-of-the art algorithms from Section 3.1 and their hybridization with the set-based DE and the similarity-based DE approaches proposed in Section 3.2 and Section 3.3, respectively. This study focuses on the smallest problem instances (in term of volume to monitor, i.e. in term of points to cover) in order to investigate which hybridization is the most appropriate to solve the optimal camera placement problem defined in this paper. The second one compares more specifically the best proposed hybridization with the best state-of-the art algorithm on a larger set of instances in order to analyze the type of situations where it can get better results.

3.4.1 Comparison of the proposed algorithms

A first experimental study has been performed, where the above-mentioned algorithms were used to solve the 12 smallest problem instances (1 to 6, and 9 to 14) defined in Tables 1 and 2. For these experimentations, all the corresponding programs are written in C/C++ and executed with a time limit of 1 000 seconds, on a computer with an Intel Core i5-3330 processor (3.00GHz) and 4 GB of RAM.

Regarding the algorithms of Section 3.1, the first one is implemented by calling CPLEX 12.7.0 with the help of ILOG Concert Technology. Here, CPLEX optimizer is set up in order to use only one single thread: it implies that the algorithm is deterministic and runs sequentially (IBM, 2017b). This setting allows a fair comparison with the other tested algorithms, and only one run per instance is needed for comparison. The greedy algorithm is also deterministic and thus only one run per instance has been performed. The third algorithm is RWLS: since it uses random numbers (as depicted in Table 4), 30 runs per instance have been performed in order to see its average behavior. Results of these three algorithms (CPLEX, Greedy, RWLS) are reported in Table 7.

Table 6: Outline of DEsim for the optimal camera placement problem.

	Input: The set C of possible camera locations. The set P of points to be covered. $\forall c \in C, cov(c) = \{p \in P: c \text{ covers } p\}$.
	Output: A set S_{best} of camera locations that covers P, i.e. such that $\bigcup_{c \in S_{best}} cov(c) = P$.
1	Generate a population Pop of random feasible solutions
2	Initialize S_{best} with the best solution of Pop
3	Set to 0 the counter cnt of generations without improvement
4	**While** the stopping condition is not met **do**
5	**If** $cnt = g$ **then**
6	**For** each individual Pop_i **do**
7	Replace Pop_i by a new random feasible solution with probability ρ
8	**End for**
9	**End if**
10	**For** each individual Pop_i **do**
11	Randomly select Pop_{r_1}, Pop_{r_2} and Pop_{r_3} such that $r_1 \neq r_2 \neq r_3 \neq i$
12	**For** each camera location j in Pop_{r_1} **do**
13	Select Pop_{r_2,u_j} and Pop_{r_3,v_j} according to similarity-based approach
14	Generate $Mut_{i,j}$ by following Equation 12
15	**End for**
16	Generate a trial solution T_i by solving the optimal camera placement subproblem where $C = Pop_i \cup Mut_i$
17	Update Pop_i and eventually S_{best} if T_i is better
18	Update cnt
19	**End for**
20	**End while**

For this experimental study, the three previous algorithms (CPLEX, Greedy, RWLS) have also been used as crossover operators of the DE algorithms presented in Sections 3.2 and 3.3 (DEset and DEsim, respectively).

CPLEX and RWLS algorithms are set up with a 10 seconds time limit. It means that either CPLEX can solve the sub-problem within 10 seconds and returns the corresponding optimal solution, or CPLEX returns a feasible solution that is not optimal. Regarding RWLS, it means that the algorithm always runs 10 seconds and it returns the best solution found so far. On the contrary, no time limit is given for the greedy algorithm, since it can not provide a feasible solution before it ends. Anyway, it never needs more than 10 seconds to solve the full problem, even the largest instance (see Table 7). Thus

the greedy algorithm should not take longer than 10 seconds to solve the sub-problem when used as a crossover operator.

Table 7: Results and statistics for CPLEX, Greedy and RWLS (best results are depicted in bold font).

	CPLEX				Greedy		RWLS		
Instance	Solution	Lower bound	Gap	Time	Solution	Time	Mean	Best	STD
1	**7**	7.00	0.00%	1.11	9	0.01	**7.00**	7	0.00
2	**4**	4.00	0.00%	0.33	**4**	0.02	**4.00**	4	0.00
3	**3**	3.00	0.00%	5.30	4	0.13	**3.00**	3	0.00
4	**5**	5.00	0.00%	10.58	7	1.06	**5.00**	5	0.00
5	**7**	7.00	0.00%	330.94	11	2.74	7.03	7	0.18
6	16 732	0.00	100.00%	-	15	6.22	**10.53**	10	0.51
9	21	17.02	18.97%	1 000.00	24	0.01	**20.00**	20	0.00
10	71	52.21	26.47%	1 000.09	83	0.11	**67.77**	67	0.63
11	17 032	0.00	100.00%	1 000.59	173	0.53	**151.67**	150	0.99
12	30 712	0.00	100.00%	1 000.15	299	1.75	**270.63**	267	1.69
13	48 392	0.00	100.00%	1 000.63	441	4.32	**428.33**	422	3.12
14	70 072	0.00	100.00%	1 001.14	641	9.16	**626.80**	619	3.91

From Table 5 and Table 6, DEset and DEsim need a random feasible solution generator: Table 8 shows the method used in this experimentation. In addition to that, the parameter settings of DEset and DEsim are given in Table 9.

Table 8: Random feasible solution generation.

Input : The set C of possible camera locations.
The set P of points to be covered.
$\forall c \in C, cov(c) = \{p \in P: c \text{ covers } p\}$.
Output : A set S of camera locations that covers P, i.e. such that $\bigcup_{c \in S} cov(c) = P$.

1 $S = \emptyset$
2 **While** $\bigcup_{c \in S} cov(c) \neq P$ **do**
3 Randomly select an uncovered point p
4 Randomly select an unused camera location c that covers p
5 $S = S \cup \{c\}$
6 **End while**

Table 9: Parameter settings for DEset and DEsim.

Parameter	DEset	DEsim
Population size	20	20
Scaling factor F	1	0.6
Number g of allowed generations without improvement	-	50
Probability ρ for an individual to be randomly regenerated	-	$\frac{1}{3}$

The results of the 6 proposed hybridizations are given in Table 10 and Table 11 (based on 30 runs per instance for each algorithm).

Table 10: Results and statistics for DEset-CPLEX, DEset-Greedy and DEset-RWLS (best results are depicted in bold font).

	Deset-CPLEX			DEset-Greedy			DEset-RWLS		
Instance	Mean	Best	STD	Mean	Best	STD	Mean	Best	STD
1	**7.00**	7	0.00	**7.00**	7	0.00	**7.00**	7	0.00
2	**4.00**	4	0.00	**4.00**	4	0.00	**4.00**	4	0.00
3	**3.00**	3	0.00	**3.00**	3	0.00	**3.00**	3	0.00
4	**5.00**	5	0.00	5.27	5	0.45	5.03	5	0.18
5	**7.93**	7	0.25	8.53	8	0.51	8.00	8	0.00
6	**11.03**	11	0.18	13.17	12	0.46	11.70	11	0.47
9	**20.47**	20	0.51	23.70	22	0.70	21.33	21	0.48
10	76.50	74	1.22	97.93	93	1.68	**75.53**	74	0.68
11	183.63	179	3.08	212.73	208	2.16	**172.40**	170	1.48
12	566.80	558	5.24	371.77	365	2.81	**326.53**	317	2.66
13	876.00	864	6.18	571.60	564	3.66	**518.07**	510	2.74
14	1 247.33	1 226	8.35	818.83	812	3.59	**748.30**	737	3.46

Table 11: Results and statistics for DEsim-CPLEX, DEsim-Greedy and DEsim-RWLS (best results are depicted in bold font).

	DEsim-CPLEX			DEsim-Greedy			DEsim-RWLS		
Instance	Mean	Best	STD	Mean	Best	STD	Mean	Best	STD
1	**7.00**	7	0.00	7.10	7	0.31	**7.00**	7	0.00
2	**4.00**	4	0.00	**4.00**	4	0.00	**4.00**	4	0.00
3	**3.00**	3	0.00	**3.00**	3	0.00	**3.00**	3	0.00
4	**5.00**	5	0.00	5.27	5	0.45	**5.00**	5	0.00
5	**7.53**	7	0.51	7.83	7	0.46	8.00	8	0.00
6	**10.97**	10	0.18	11.77	11	0.43	11.93	11	0.25
9	**21.07**	21	0.25	24.30	23	0.88	21.27	21	0.45
10	**70.97**	70	0.67	99.43	97	1.91	73.60	72	0.97
11	**152.13**	149	1.41	230.07	217	6.78	163.87	159	2.43
12	**268.27**	262	4.47	413.00	408	2.24	318.23	293	10.98
13	**424.47**	414	5.36	638.70	629	5.10	532.80	520	5.48
14	**612.33**	600	6.59	912.30	902	5.07	790.70	772	6.50

From Table 7, one can see that CPLEX finds the optimal solution within the time limit for instances 1 to 5. For instance 6, no runtime is reported: CPLEX stops because it needs more RAM than available on the computer. By default, the solution is thus set to the number of possible camera locations in the reduced instance (taken from Table 2). Regarding the second group of instances (9 to 14), CPLEX

gets decent solutions only for instances 9 and 10. For instances 11 to 14, CPLEX provides no better solution than the number of possible camera locations available in the reduced instance. These results clearly show that CPLEX can not be used to solve large instances within the given time limit. On the contrary, Greedy succeeds to find a solution for all instances. It is fast whatever the size of the instance, but it gives poor results in comparison with RWLS, which beats Greedy and CPLEX on 11 instances.

From Table 10, DEset-CPLEX is slightly better than DEset-RWLS when solving instances 1 to 6 and instance 9. However, when the size of the solutions increases (as in instances 10 to 14), DEset-RWLS outperforms DEset-CPLEX. The main reason is that the XOR operation in Equation 11 is not so helpful for the considered problem: the probability of having exactly the same camera location in Pop_{r_1} and Pop_{r_2} is very low, thus the XOR operation removes only a few camera locations. As a consequence, it can not reduce the size of $Pop_i \cup Mut_i$, which means that the subproblem in the crossover is harder to solve within the given time limit: CPLEX do not have enough time to find interesting solutions. On the contrary, RWLS is faster and is able to reach better solutions for the subproblem, which leads to the final better results of DEset-RWLS. Not surprisingly, DEset-Greedy can not compete with DEset-RWLS. However, as it is fast compared to CPLEX, it outperforms DEset-CPLEX as soon as the size of the solutions increases (instances 12 to 14).

From Table 11, DEsim-CPLEX clearly outperforms the other hybridizations. Compared to DEsim-RWLS, the gap grows up to about 22.5% for instance 14 (according to the reported mean values). The similarity-based approach allows to reduce the size of $Pop_i \cup Mut_i$, which makes the subproblem easier to solve for CPLEX.

When comparing the results of Table 7 and Table 10, it can be observed that CPLEX clearly benefits from the hybridization with DEset: obviously, the subproblem in the crossover is much smaller than the full instance, and thus it is easier to solve. DEset-Greedy succeeds in improving the results of Greedy for instances 1 to 6 and for instance 9, but it fails for the largest instances (10 to 14). Regarding RWLS, no improvement is achieved by using DEset-RWLS.

When focusing on Table 10 and Table 11, it turns out that the DEsim-CPLEX hybridization benefits most from the DEsim approach with more than 50% of improvement (with regard to DEset-CPLEX) on the largest instances (12 to 14), and it is equivalent to or better than DEset-CPLEX for all instances except instance 9. Moreover DEsim-CPLEX outperforms DEset-RWLS for all instances. In the meantime, DEsim-Greedy gets not convincing results: it improves the results of DEset-Greedy for instances 5 and 6, but it is worse regarding instances 1 and 9 to 14. It is the same for DEsim-RWLS, which improves the results of DEset-RWLS for 5 instances (4 and 9 to 12) and fails for 3 other instances (6, 13 and 14).

Now, from Table 7 and Table 11, DEsim-Greedy provides equivalent or better results compared to Greedy for instances 1 to 6, but it is not successful for instances 9 to 14. Similar results are observed for DEsim-RWLS against RWLS: the latter wins for instances 5 and 6 and for instances 9 to 14 while the former is equivalent only for instances 1 to 4. However, DEsim-CPLEX is equivalent or competitive with regard to RWLS for instances 1 to 6, and 9 to 11. And it is worth noting that DEsim-CPLEX outperforms RWLS for the largest instances (12 to 14).

Finally, a global comparison between CPLEX, Greedy, RWLS, and their hybridizations with DEset and DEsim, is presented in Table 12. In this table, statistical significance is tested using the Kruskal-Wallis statistical test at 95% confidence level followed by Fisher's least significant difference post hoc test. The results that are significantly better than the ones of the other algorithms, according to this statistical test, are preceded with a star symbol. As one can see, RWLS obtains the best results for all the considered small instances. However, DEsim-CPLEX achieves a similar performance for instances 1 to 4, instance 6, and instances 11 to 14, i.e. no significant difference is found between RWLS and DEsim-CPLEX for these instances. It is especially interesting to notice that these two algorithms are the only ones to obtain significantly better results than the others for instances 11 to 14, which are the largest among the considered ones.

Table 12: Results for all tested algorithms (best results are depicted in bold font, and a star denotes the results that are significantly better than the others according to the Kruskal-Wallis statistical test at 95% confidence level followed by Fisher's least significant difference post hoc test).

	CPLEX	Greedy	RWLS	DEset-CPLEX	DEset-Greedy	DEset-RWLS	DEsim-CPLEX	DEsim-Greedy	DEsim-RWLS
Inst.	Solution	Solution	Mean	Mean	Mean	Mean	Mean	Mean	Mean
1	* **7**	9	* **7.00**	* **7.00**	* **7.00**	* **7.00**	* **7.00**	* 7.10	* **7.00**
2	* **4**	* **4**	* **4.00**	* **4.00**	* **4.00**	* **4.00**	* **4.00**	* **4.00**	* **4.00**
3	* **3**	4	* **3.00**	* **3.00**	* **3.00**	* **3.00**	* **3.00**	* **3.00**	* **3.00**
4	* **5**	7	* **5.00**	* **5.00**	5.27	* 5.03	* **5.00**	5.27	* **5.00**
5	* **7**	11	* 7.03	7.93	8.53	8.00	7.53	7.83	8.00
6	16 732	15	* **10.53**	* 11.03	13.17	11.70	* 10.97	11.77	11.93
9	21	24	* **20.00**	* 20.47	23.70	21.33	21.07	24.30	21.27
10	71	83	* **67.77**	76.50	97.93	75.53	70.97	99.43	73.60
11	17 032	173	* **151.67**	183.63	212.73	172.40	* 152.13	230.07	163.87
12	30 712	299	* 270.63	566.80	371.77	326.53	* **268.27**	413.00	318.23
13	48 392	441	* 428.33	876.00	571.60	518.07	* **424.47**	638.70	532.80
14	70 072	641	* 626.80	1 247.33	818.83	748.30	* **612.33**	912.30	790.70

This first experimentation points out that RWLS is the best algorithm for solving the considered instances. Moreover, DEsim-CPLEX is the best proposed hybridization. It seems also competitive with RWLS when the size of the problem increases, and the next section is devoted to a more detailed comparison of these two algorithms.

3.4.2 Comparison of RWLS and DEsim-CPLEX

A second experimental study has been performed by solving the largest problem instances (7 to 8, and 15 to 18) with RWLS and DEsim-CPLEX. This setting allows to compare these algorithms on the whole set of problem instances defined in Tables 1 and 2. For these additional experimentations, the runtime limit depends on the size of the problem instance in the following way. The considered large instances have been solved first by using Greedy (results are reported in Table 13), then it is decided that $100 \times \lceil t_i \rceil$ seconds are allowed for each run, where t_i refers to the runtime of Greedy when solving instance i. This setting is similar to the one observed for instance 14 in Section 3.4.1 (i.e. 9.16 seconds for Greedy, and a time limit of 1 000 seconds). Regarding DEsim-CPLEX, the time limit for the crossover is set to $\lceil t_i \rceil$, which is also similar to the setting defined in Section 3.4.1 (where 10 seconds

are allowed for the crossover, i.e. 1% of the allowed total runtime). The results of RWLS and DEsim-CPLEX are given in Table 13 (based on 30 runs per instance for each algorithm). In Table 13, results for instance 1 to 6 and 9 to 14 are taken from Table 7 and Table 11.

Table 13: Results for Greedy, RWLS and DEsim-CPLEX (best results are depicted in bold font, and a star denotes which algorithm, between DEsim-CPLEX and RWLS, significantly outperforms the other according to the Wilcoxon-Mann-Whitney statistical test at 95% confidence level).

	Greedy		RWLS			DEsim-CPLEX			
Instance	Solution	Time	Mean	Best	STD	Mean	Best	STD	Gap (%)
1	9	0.01	* **7.00**	7	0.00	* **7.00**	7	0.00	0.00
2	**4**	0.02	* **4.00**	4	0.00	* **4.00**	4	0.00	0.00
3	4	0.13	* **3.00**	3	0.00	* **3.00**	3	0.00	0.00
4	7	1.06	* **5.00**	5	0.00	* **5.00**	5	0.00	0.00
5	11	2.74	* **7.03**	7	0.18	7.53	7	0.51	7.11
6	15	6.22	* **10.53**	10	0.51	10.97	10	0.18	4.11
7	25	18.40	* **18.07**	17	0.45	19.50	19	0.51	7.93
8	38	46.21	* **28.69**	28	0.65	30.80	30	0.41	7.32
9	24	0.01	* **20.00**	20	0.00	21.07	21	0.25	5.33
10	83	0.11	* **67.77**	67	0.63	70.97	70	0.67	4.72
11	173	0.53	* **151.67**	150	0.99	* 152.13	149	1.41	0.31
12	299	1.75	270.63	267	1.69	* **268.27**	262	4.47	-0.87
13	441	4.32	428.33	422	3.12	* **424.47**	414	5.36	-0.90
14	641	9.16	626.80	619	3.91	* **612.33**	600	6.59	-2.31
15	1 139	30.56	1 108.70	1 095	6.79	* **1 061.93**	1 043	7.60	-4.22
16	1 748	75.16	1 723.33	1 710	7.83	* **1 621.27**	1 601	13.57	-5.92
17	2 498	156.44	2 482.93	2 468	8.16	* **2 299.47**	2 277	12.26	-7.39
18	3 415	290.31	3 393.50	3 352	20.62	* **3 127.67**	3 104	19.48	-7.83

The mean values in Table 13 show that both algorithms are equivalent for the smallest instances of the first class (1 to 4). Moreover, RWLS performs better for the largest instances of the first class (5 to 8) and for the smallest instances of the second class (9 to 11). However, DEsim-CPLEX wins for the 7 largest instances of the second class (12 to 18) and the percentage gap increases with the size of the problem instance (see last column of Table 13, and Figure 4).

In addition to that, the Wilcoxon-Mann-Whitney statistical test is used, at 95% confidence level, to determine which algorithm, between DEsim-CPLEX and RWLS, obtains significantly better results than the other. In Table 13, for each instance, the result obtained by the best performing algorithm is preceded with a star symbol. If the results of both algorithms are preceded with a star, then no significant difference is found for the corresponding instance. One can see that RWLS significantly outperforms DEsim-CPLEX for instances 5 to 10. However, for the largest instances 12 to 18, the opposite situation occurs, i.e. DEsim-CPLEX significantly outperforms RWLS, which confirms the above analysis of the reported mean values.

From this second experimentation, it can be concluded that RWLS is best suited when an operational need of 100 pixels per meter is needed (instances 1 to 8), whereas DEsim-CPLEX is best suited for

large problems with an operational need of 500 pixels per meter (instances 9 to 18). In term of USCP, DEsim-CPLEX seems more appropriate when the zero-one input matrix (see Section 2.3 and Table 2) is sparser and when the number of ones per row is lower. But this observation can not extend to general USCP benchmark problems, since the DEsim approach uses a context-dependent information of the given sets (i.e. the similarity between the camera locations), which is not available in case of general USCP.

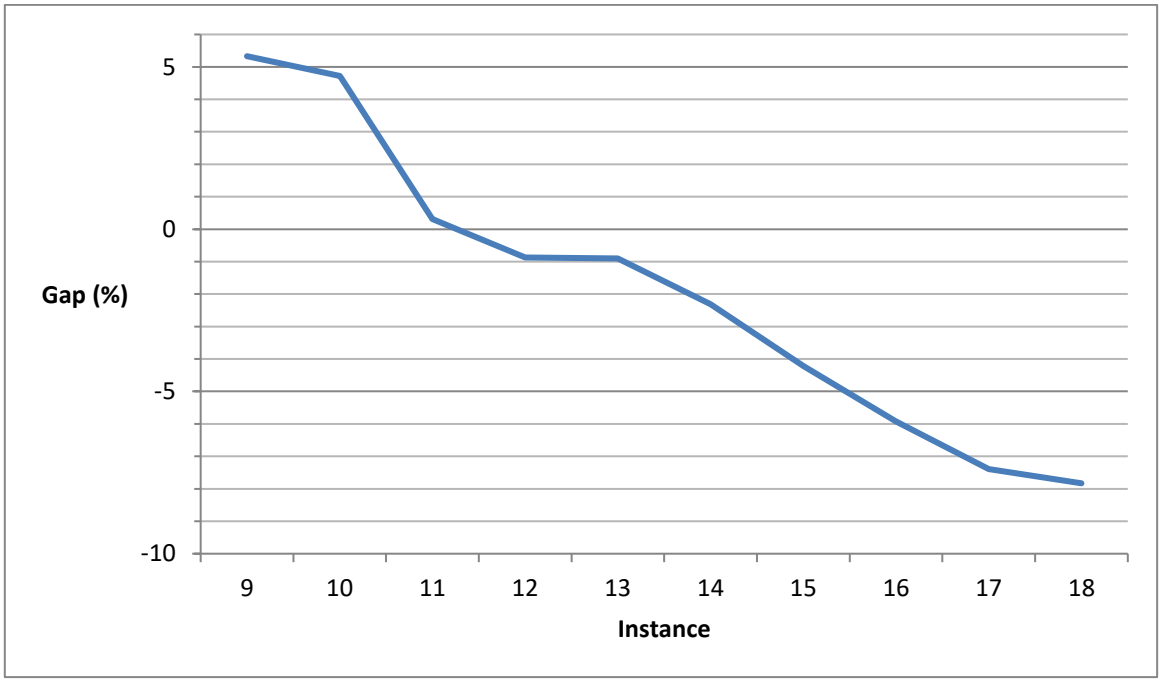

Figure 4: Evolution of the gap percentage between DEsim-CPLEX and RWLS for instances 9 to18.

4 Conclusion

This paper deals with the optimal camera placement problem with the constraint that a full three-dimensional coverage of the monitored area is needed.

After explaining in detail the problem modelling, this optimization problem is stated as a unicost set covering problem (USCP). Then, 18 instances inspired by real-world applications are provided in order to investigate the efficiency of the proposed algorithms.

The aim of this work is to estimate the benefit of the differential evolution (DE) paradigm in the context of this combinatorial optimization problem. A selection of 3 state-of-the-art algorithms (CPLEX, Greedy and RWLS) is presented, and 2 differential evolution approaches (DEset and DEsim) are proposed for the purpose of hybridization. The main contribution consists in the design of DEsim, which is a simple similarity-based approach that allows to make sense of the DE mutation scheme for the considered optimization problem.

An experimental study has been performed in order to compare all these algorithms when solving the considered instances. The reported results show that RWLS and DEsim-CPLEX are the most interesting. Each of them can find better results on different class of problem instances, depending on the operational need and, thus also, on the nature of the input visibility matrix.

A first perspective is to achieve a comprehensive study of the influence of the algorithm parameters. For example, regarding the DEset approach, the impact of the scaling factor strategies from the literature (Prado et al., 2010) can be determined. Then, the DEsim can be certainly fine-tuned by testing a large set of values for F, g and ρ. And for both approaches, other stopping conditions can be proposed when using RWLS in the crossover operator. It can also be considered to add self-adaptive techniques so that the proposed algorithms will be less user-dependent and will potentially give better results. Since the similarity-based DE approach takes advantage of the real nature of the sets (camera locations, here), another perspective would be to examine to what extent it can help to improve the solution quality in other real-world applications that can be stated as USCP. In future work, it can also be planned to try other hybridizations by considering other exact methods or local search heuristics when solving the subproblem given in the crossover operator: it will be interesting to determine which method is the most appropriate for such an hybridization. Another direction would be to consider different coverage models from the literature (Mavrinac and Chen, 2013) in order to investigate the efficiency of the proposed methods for other practical applications, such as inspection or measurement of industrial products.

Notes

[1] http://www.mage.fst.uha.fr/brevilliers/data/

References

Axis (2017), "Pixel density", available at: https://www.axis.com/us/en/learning/web-articles/perfect-pixel-count/pixel-density (accessed 8 September 2017).

Beasley, J.E. (1990), "OR-Library: Distributing Test Problems by Electronic Mail", *Journal of the Operational Research Society*, Vol. 41 No. 11, pp. 1069-1072.

Chi, Z., Xuan, J., Ren, Z., Xie, X. and Guo H. (2017), "Multi-Level Random Walk for Software Test Suite Reduction", *IEEE Computational Intelligence Magazine*, Vol. 12 No. 2, pp. 24-33.

Chrysostomou, D. and Gasteratos, A. (2012), "Optimum multi-camera arrangement using a bee colony algorithm", in *2012 IEEE International Conference on Imaging Systems and Techniques, Manchester, UK, 16-17 July 2012*, IEEE, pp. 387-392.

Chvatal, V. (1979), "A Greedy Heuristic for the Set-Covering Problem", *Mathematics of Operations Research*, Vol. 4 No. 3, pp. 233-235.

Das, S., Mullick, S.S. and Suganthan, P.N. (2016), "Recent Advances in Differential Evolution – An Updated Survey", *Swarm and Evolutionary Computation*, Vol. 27, pp. 1-30.

David, P., Idasiak, V. and Kratz F. (2007), "A Sensor Placement Approach for the Monitoring of Indoor Scenes", in Kortuem, G., Finney, J., Lea, R. and Sundramoorthy V. (Eds), *European Conference on Smart Sensing and Context (EuroSSC 2007), Kendal, England, 23-25 Oct. 2007*, Springer, Berlin, pp. 110-125.

Demirović, E., Le Calvar, T., Musliu, N. and Inoue, K. (2016), "An Exact Algorithm for Unicost Set Covering", paper presented at the Doctoral Program of the 22nd International Conference on the Principles and Practice of Constraint Programming (CP 2016), 5-9 September 2016, Toulouse, France, available at:
https://www.dbai.tuwien.ac.at/user/demir/papers/An%20Exact%20Algorithm%20for%20Unicost%20Set%20Covering.pdf
(accessed 12 September 2017).

Ercan, A.O., Yang, D.B., El Gamal A. and Guibas L.J. (2006), "Optimal Placement and Selection of Camera Network Nodes for Target Localization", in Gibbons, P.B., Abdelzaher, T., Aspnes J. and Rao R. (Eds), *International Conference on Distributed Computing in Sensor System (DCOSS 2006), San Francisco, CA, USA, 18-20 June 2006*, Springer, Berlin.

Fulkerson, D.R., Nemhauser, G.L. and Trotter, L.E. (1974), "Two Computationally Difficult Set Covering Problems that Arise in Computing the 1-width of Incidence Matrices of Steiner Triple Systems", in Balinski, M.L. (Ed.), *Approaches to Integer Programming*, Springer, Berlin, Heidelberg, pp. 72-81.

Gao, C., Yao, X., Weise, T. and Li, J. (2015), "An Efficient Local Search Heuristic with Row Weighting for the Unicost Set Covering Problem", *European Journal of Operational Research*, Vol. 246 No. 3, pp. 750-761.

Gordon, C.L., Blackwell, C.L., Bradtmiller, B., Parham, J.L., Barrientos, P., Paquette, S.P., Corner, B.D., Carson, J.M., Venezia, J.C., Rockwell, B.M., Mucher, M. and Kristensen, S. (2014), "2012 Anthropometric Survey of U.S. Army Personnel: Methods and Summary Statistics", Technical Report, NATICK/TR-15/007, U.S. Army Natick Soldier Research, Development and Engineering Center, Natick, MA, USA.

Horster, E. and Lienhart, R. (2009), "Optimal Placement of Multiple Visual Sensors", in Aghajan, H. and Cavallaro, A. (Ed.), *Multi-Camera Networks: Principles and Applications*, Elsevier, pp. 117-138.

Huang, P. and Wang Y. (2008), "The Impact of Changing Resolutions on Face Recognition", *2008 International Workshop on Education Technology and Training & 2008 International Workshop on Geoscience and Remote Sensing, Shanghai, China, 21-22 Dec. 2008*, IEEE, pp. 622-625.

IBM (2017a), "CPLEX Optimizer", available at:
https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/
(accessed 12 September 2017).

IBM (2017b), "global thread count", available at:
https://www.ibm.com/support/knowledgecenter/SSSA5P_12.7.0/ilog.odms.cplex.help/CPLEX/Parameters/topics/Threads.html
(accessed 12 September 2017).

Johnson, D.S. (1974), "Approximations Algorithms for Combinatorial Problems", *Journal of Computer and System Sciences*, Vol. 9 No. 3, pp. 256-278.

Konda, K.R. and Conci, N. (2013), “Global and local coverage maximization in multi-camera networks by stochastic optimization”, *Infocommunications Journal*, Vol. 5 No. 1, pp. 1-8.

Liang, Y., Li, C.T., Guan, Y. and Hu, Y. (2016), “Gait Recognition Based on the Golden Ratio”, *EURASIP Journal on Image and Video Processing*, 2016:22.

Liu, J., Sridharan, S. and Fookes, C. (2016), “Recent Advances in Camera Planning for Large Area Surveillance: A Comprehensive Review”, *ACM Computing Surveys (CSUR)*, Vol. 49 No. 1.

Liu, J., Sridharan, S., Fookes, C. and Wark T. (2014), “Optimal Camera Planning Under Versatile User Constraints in Multi-Camera Image Processing Systems”, *IEEE Transactions on Image Processing*, Vol. 23 No. 1, pp. 171-184.

Mahmood, Z., Ali, T. and Khan, S.U. (2016), “Effects of Pose and Image Resolution on Automatic Face Recognition”, *IET Biometrics*, Vol. 5 No. 2, pp. 111-119.

Maravilha, A.L., Ramirez, J.A. and Campelo, F. (2013), “A New Algorithm Based on Differential Evolution for Combinatorial Optimization”, in *2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence (BRICS-CCI & CBIC), Ipojuca, Brazil, 8-11 Sept. 2013*, IEEE.

Maravilha, A.L., Ramirez, J.A. and Campelo, F. (2014), “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),Vancouver, BC, Canada, 12-16 July 2014*.

Marciniak, T., Chmielewska, A., Weychan, R., Parzych, M. and Dabrowski, A. (2015), “Influence of Low Resolution of Images on Reliability of Face Detection and Recognition”, *Multimedia Tools and Applications*, Vol. 74 No. 12, pp 4329-4349.

Mavrinac, A. and Chen, X. (2013), “Modeling Coverage in Camera Networks: A Survey”, *International Journal of Computer Vision*, Vol. 101 No. 1, pp. 205-226.

Miyazaki, N., Tsuji, K., Zheng, M., Nakashima, M., Matsuda, Y. and Segawa, E. (2015), “Privacy-conscious human detection using low-resolution video”, *3rd IAPR Asian Conference on Pattern Recognition (ACPR), Kuala Lumpur, Malaysia, 3-6 Nov. 2015*, IEEE, pp. 326-330.

Morsly, Y., Aouf, N., Djouadi, M.S. and Richardson, M. (2012), “Particle Swarm Optimization Inspired Probability Algorithm for Optimal Camera Network Placement”, *IEEE Sensors Journal*, Vol. 12 No. 5, pp. 1402-1412.

Prado, R.S., Silva, R.C.P., Guimarães, F.G. and Neto, O.M. (2010), “Using Differential Evolution for Combinatorial Optimization: A General Approach”, *2010 IEEE International Conference on Systems Man and Cybernetics (SMC), Istanbul, Turkey, 10-13 Oct. 2010*, IEEE.

Rebai, M., Le Berre, M., Hnaien, F. and Snoussi, H. (2016), “Exact Biobjective Optimization Methods for Camera Coverage Problem in Three-Dimensional Areas”, *IEEE Sensors Journal*, Vol. 16 No. 9, pp. 3323-3331.

Ryerkerk, M.L., Averill, R.C., Deb, K. and Goodman, E.D. (2017), "Solving Metameric Variable-Length Optimization Problems Using Genetic Algorithms", *Genetic Programming and Evolvable Machines*, Vol. 18 No. 2, pp. 247-277.

Storn, R. and Price, K. (1997), "Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces", *Journal of Global Optimization*, Vol. 11 No. 4, pp. 341-359.

Talbi, E.G. (2002), "A Taxonomy of Hybrid Metaheuristics", *Journal of Heuristics*, Vol. 8 No. 5, pp. 541-564.

Van den Hengel, A., Hill, R., Ward, B., Cichovski, A, Detmold, H., Madden, C., Dick, A. and Bastian, J. (2009), "Automatic camera placement for large scale surveillance networks", in *2009 Workshop on Applications of Computer Vision (WACV), Snowbird, UT, USA, 7-8 Dec. 2009*, IEEE.

Yelbay, B., Birbil S.I. and Bülbül, K. (2015), "The Set Covering Problem Revisited: An Empirical Study of the Value of Dual Information", *Journal of Industrial and Management Optimization*, Vol. 11 No. 2, pp. 575-594.

Zhang, B., Zhang, X., Chen, X. and Fang , Y. (2016), "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), Banff, AB, Canada, 12-15 July 2016*, pp. 1246-1251.

Zhang, H., Xia, L., Wang, P., Cui, J., Tang, C., Deng, N. and Ma, N. (2013) "An Optimized Placement Algorithm for Collaborative Information Processing at a Wireless Camera Network", *2013 IEEE International Conference on Multimedia and Expo (ICME), San Jose, CA, USA, 15-19 July 2013*, IEEE, pp. 1-6.

Zhang, J., Pu, J., Chen, C. and Fleischer, R. (2010), "Low-Resolution Gait Recognition", *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, Vol. 40 No. 4, pp. 986-996.

Zhang, X., Chen, X., Alarcon-Herrera, J.L. and Fang, Y. (2015), "3-D Model-Based Multi-Camera Deployment: A Recursive Convex Optimization Approach", *IEEE/ASME Transactions on Mechatronics*, Vol. 20 No. 6, pp 3157-3169.

Zhao, J. (2011), "Camera Planning and Fusion in a Heterogeneous Camera Network", Unpublished Manuscript, Ph.D. dissertation, University of Kentucky, USA.

Zhuang, Z., Landsittel, D., Benson, S., Roberge, R. and Shaffer, R. (2010), "Facial Anthropometric Differences among Gender, Ethnicity, and Age Groups", *The Annals of Occupational Hygiene*, Vol. 54 No. 4, pp. 391-402.