Probabilistic Sensing Model for Sensor Placement Optimization based on Line-of-sight Coverage

Vahab Akbarzadeh Christian Gagné Marc Parizeau Meysam Argany
Mir Abolfazl Mostafavi

November 1, 2012

Abstract

This paper proposes a probabilistic sensor model for the optimization of sensor placement. Traditional schemes rely on simple sensor behaviours and environmental factors. The consequences of these oversimplifications are unrealistic simulation of sensor performance and, thus, suboptimal sensor placement. In this paper, we develop a novel probabilistic sensing model for sensors with line-of-sight based coverage (e.g. cameras) to tackle the sensor placement problem for these sensors. The probabilistic sensing model consists of membership functions for sensing range and sensing angle, which takes into consideration sensing capacity probability as well as critical environmental factors such as terrain topography. We then implement several optimization schemes for sensor placement optimization, including simulated annealing, L-BFGS, and CMA-ES.

1 Introduction

Wireless Sensor Networks (WSN) are built from a collection of small, inexpensive sensor devices, where each sensor has limited sensing, storage, processing, and communication capabilities. With the recent proliferation of Micro-Electro-Mechanical Systems (MEMS), we have seen a rapid increase of interest in WSNs [30], where sensors can make measurements in the environment and gather information for end-users.

There are a number of fundamental issues that should be addressed for effective exploitation of WSNs, such as localization, tracking, security, data aggregation, and placement. Placement is an example of a more general problem of configuring sensor parameters. Depending on the application of WSN and the sensor type being used, each sensor has a number of variable parameters that must be determined, e.g., latitude and longitude, orientation, and operating range of each sensor in the placement problem. There are four main issues which should be taken into consideration for an optimal placement of WSNs, namely, performance maximization, reliability maximization, energy saving, and cost minimization.

Considering a region of interest monitored by sensors, overall performance of the network is measured by coverage [14, 16, 19, 26, 28, 31]. In general, one of the basic requirements for a WSN is that each location in a region of interest should be within the sensing range of at least one of the sensors. An alternative approach is to have a region of interest covered simultaneously by at least K sensors [28, 31].

Although many deterministic methods have been explored to address the problem of coverage, traditional sensor placement strategies often rely on oversimplified sensor models and environmental factors [23,25,26,28,31]. These deterministic approaches cannot deal with environmental factors such as terrain topography, and usually assume an omnidirectional disk sensing model for each sensor. In fact, under the assumptions of uniform disk sensing model, it has been shown that optimal coverage can be deterministically achieved with a regular placement of sensors [5,14,16]. Similar results have also been reported when multiple coverage of the target area is required [5, 26, 28, 31].

The direct consequence of such oversimplifications is that the theoretical perfect coverage shown in deterministic methods may not hold true in practice. This may be the result of a number of causes. First, most sensor placement optimization methods assume that sensors are placed on a 2D plane, without taking into account the topography of the terrain [5, 14, 16]. Second, many methods assume that sensors have omnidirectional sensing capabilities [15]. But antennas and microphones have non-uniform 3D reception fields that depend on factors like orientation, distance, and other environmental factors [15], cameras have narrow field of views, etc. Third, sensors usually do not have a binary coverage range as it is often assumed in traditional sensor placement methods [5,16]. Although some probabilistic sensing range models [1,5,14,16,33] and sensing models with irregular sensing ranges [6] have been proposed, they all operate on a 2D flat space and are omnidirectional. Dhillon et al. [9] was among the first who proposed the combination of terrain modelling and probabilistic sensor model for sensor placement. Still, the paper implements and tests an unrealistic model for the terrain and the obstacles inside the terrain. Besides, their greedy approach has high chance of getting trapped in

V. Akbarzadeh, C. Gagné, M. Parizeau, M. Argany and M.A. Mostafavi. Probabilistic Sensing Model for Sensor Placement Optimization Based on Line-of-Sight Coverage. *IEEE Transactions on Instrumentation and Measurement,* in-press, 2012. doi: 10.1109/TIM.2012.2214952

a suboptimal solution. Ma et al. [21] has also proposed a sensor placement method based on virtual force mechanism and simulated annealing. Although authors presented their approach to have a 3D model, their model does not take topography or obstacles into consideration, also they used a binary sensing coverage model. Recently, Topcuoglu et al. [27] proposed a new formulation for deployment of the sensors in a synthetically generated 3D environment. Although the proposed approach makes several realistic assumptions regarding the modelling of the environment and sensors, it assumes a binary sensing area for each sensor inside the environment. The combinational effects of terrain variations, blind points sensing angle or irregular sensing range, and probabilistic sensing property of sensors have never been studied before. For a rather recent survey on coverage optimization algorithms for directional sensor networks, interested readers are referred to [12].

Placement problem in WSNs is closely related to the observers sitting problem which has been addressed in the geomatics science literature [10, 20]. In this problem, one tries to find the optimal position for a number of observers, required to cover a certain ratio of an area. Methods proposed for this problem have been applied to determine the location of telecommunication base stations [8], to protect endangered species [4], and to determine the location of wind turbines [17]. More recently, Murray et al. [22] combined the idea of viewshed analysis from geomatic science within a surveillance camera placement problem.

The limitations of the deterministic placement methods are thus obvious, and the 100% coverage claims are often over-estimated. This issue is critical because it further complicates the problem of sensor placement: while a WSN may seem to satisfy the requirements to achieve full coverage on a target area using a deterministic method, the deployers of such a network have no means of ensuring that this coverage is truly effective in a real environment.

Facing this challenge, we follow a more flexible non-deterministic avenue. Our aim is to optimize sensor placement using topographic information of the terrain and probabilistic sensor modelling. Our approach differs from previous methods in the following three ways:

1. Deterministic schemes only consider 2D environments and ignore the effects of elevation, whereas our method takes into account the 3D terrain information. In our approach, the environment is defined using a Geographic Information System (GIS), which is an information system designed to store, manipulate and analyze geographically referenced data [29].

2. Deterministic schemes usually assume omnidirectional sensors, whereas our method allows for constraints to be applied on sensors, such as limited sensing angles and range.

3. Deterministic schemes implement mainly binary coverage, i.e., a point can only be classified as covered or uncovered, whereas our method applies probabilistic coverage in both sensing distance and sensing angle.

In summary, deterministic approaches assume omni-directional sensors with binary coverage on a flat terrain, while our probabilistic approach assumes directional sensors, for which omni-directional sensors are a special case, with probabilistic coverage on a realistic spatial model of the environment.

In order to tackle these problems, we develop a novel probabilistic coverage function for sensor placement that takes into account the above mentioned issues, and then compare this approach with some classical optimization algorithms. This paper extends our previous work [2, 3] by proposing directional and probabilistic sensor models along the pan and tilt sensing angles, and comparing the optimization with other methods, that is simulated annealing and L-BFGS.

The remainder of the paper is organized as follows. The proposed model is presented in the next section (Sec. 2), followed by a presentation of the optimization methods in Sec. 3, including experimental protocol and results on sensor placement. We conclude the paper in section IV with a summary of results and perspectives.

2 Proposed Probabilistic Sensor Model

2.1 Coverage Definition in a Sensor Network

The sensing model mainly depends on distance, orientation, and visibility. We first assume that all sensors are positioned at a certain constant height τ above the ground level. The sensor position is thus described by a 3D point $\mathbf{p} = (x, y, z)$, where (x, y) are free parameters and $z = g(x, y)$ is constrained by the terrain elevation at position (x, y), as defined by a Digital Elevation Model (DEM) provided by a GIS. We further assume that the anisotropic properties of sensors are fully defined by a pan angle θ around the vertical axis and a tilt angle ξ around horizontal axis. Given the DEM, a sensor network $N = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n\}$ of n sensors is thus fully specified by $4n$ free parameters $\mathbf{s}_i = (\mathbf{p}_i, \theta_i, \xi_i)$, $i = 1, 2, \ldots, n$, with $\mathbf{p}_i = (x_i, y_i)$ (see Figure 1).

Now the coverage $C(\mathbf{s}_i, \mathbf{q})$ of sensor $\mathbf{s}_i$ at point $\mathbf{q}$ in the environment can be defined as a function of distance $d(\mathbf{s}_i, \mathbf{q}) = \|\mathbf{p}_i - \mathbf{q}\|$, pan angle $p(\mathbf{s}_i, \mathbf{q}) = \angle_p(\mathbf{q} - \mathbf{p}_i) - \theta_i$, tilt angle $t(\mathbf{s}_i, \mathbf{q}) = \angle_t(\mathbf{q} - \mathbf{p}_i) - \xi_i$, and visibility $v(\mathbf{s}_i, \mathbf{q})$ from the sensor:

$$C(\mathbf{s}_i, \mathbf{q}) = f[\mu_d(\|\mathbf{p}_i - \mathbf{q}\|), \mu_p(\angle_p(\mathbf{q} - \mathbf{p}_i) - \theta_i),\ \mu_t(\angle_t(\mathbf{q} - \mathbf{p}_i) - \xi_i), v(\mathbf{p}_i, \mathbf{q})], \tag{1}$$

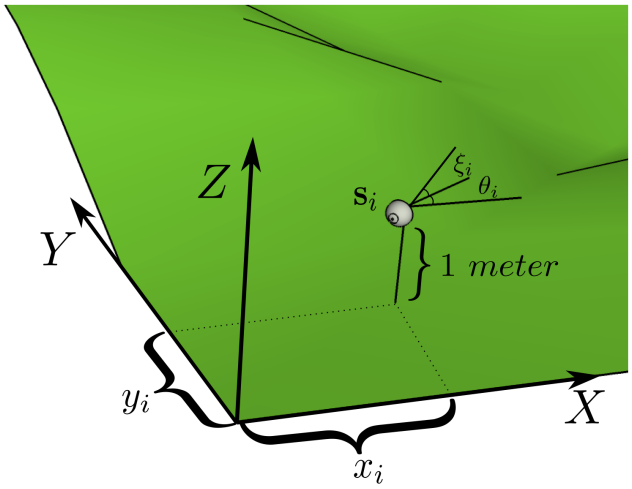


Figure 1: The free parameters $(x_i, y_i, \theta_i, \xi_i)$ for sensor $\mathbf{s}_i$ inside the environment.

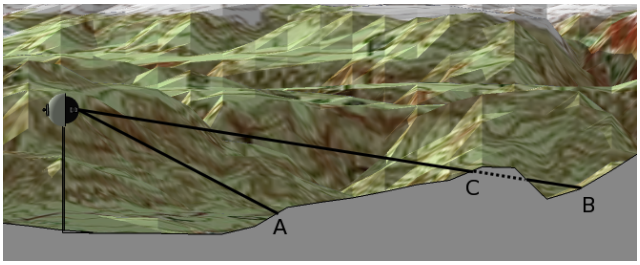


Figure 2: The effect of visibility function. Here sensor and point **A** are inter-visible, therefore $v(\mathbf{s}_i, \mathbf{A}) = 1$, but it is not the case for $\mathbf{s}_i$ and **B**, because the line-of-sight is obscured at point **C**.

where $\angle_p(\mathbf{q} - \mathbf{p}_i) = \arctan\left(\frac{y_\mathbf{q} - y_{\mathbf{p}_i}}{x_\mathbf{q} - x_{\mathbf{p}_i}}\right)$ is the angle between the sensor $\mathbf{s}_i$ and the point $\mathbf{q}$ along the **X** direction, and $\angle_t(\mathbf{q} - \mathbf{p}_i) = \arctan\left(\frac{z_\mathbf{q} - z_{\mathbf{p}_i}}{\|\mathbf{p}_i - \mathbf{q}\|}\right)$ is the angle between the sensor $\mathbf{s}_i$ and the point $\mathbf{q}$ along the **Z** direction. In other words, for $\mathbf{q}$ to be covered by sensor $\mathbf{s}_i$, we need to take into account its sensing range, sensing angles, and visibility. Let $\mu_d, \mu_p, \mu_t \in [0, 1]$ represent some membership functions of the mentioned coverage conditions, then Equation 1 can be rewritten as multiplication of these memberships:

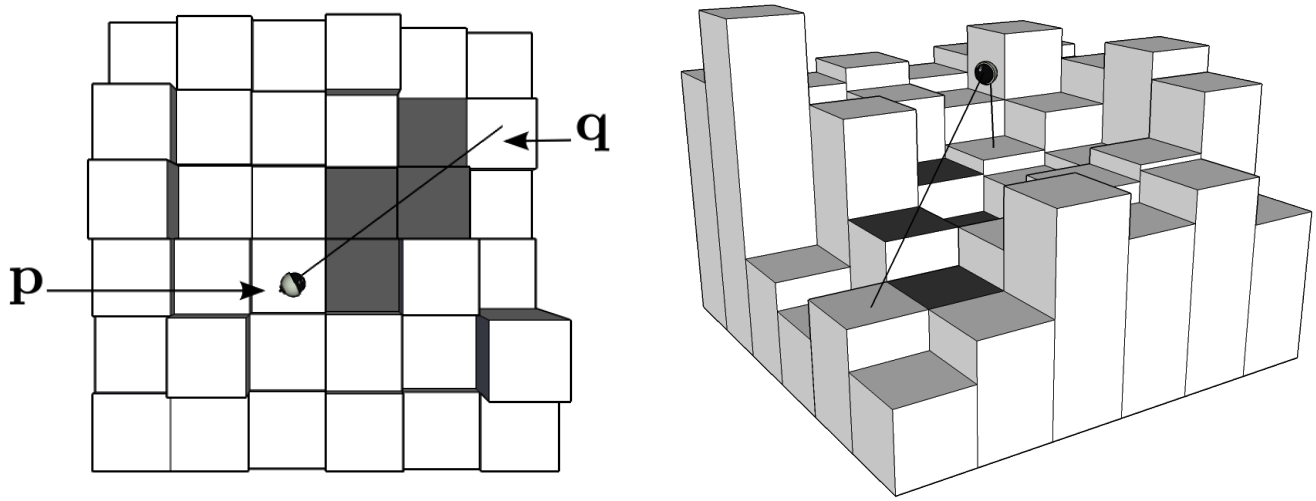
$$C(\mathbf{s}_i, \mathbf{q}) = \mu_d(\|\mathbf{p}_i - \mathbf{q}\|) \cdot \mu_p(\angle_p(\mathbf{q} - \mathbf{p}_i) - \theta_i) \cdot \mu_t(\angle_t(\mathbf{q} - \mathbf{p}_i) - \xi_i) \cdot v(\mathbf{p}_i, \mathbf{q}), \tag{2}$$

Function $v(\mathbf{p}_i, \mathbf{q})$ is usually binary. Given a sensor position $\mathbf{p}_i$, if the line-of-sight between sensor $\mathbf{s}_i$ and $\mathbf{q}$ is obscured, then we assume that the coverage cannot be met ($v = 0$), otherwise the visibility condition is fully respected ($v = 1$), (see Figure 2). In our experiment, we assume that all sensors are one metre above the ground. Memberships μ_d, μ_p, and μ_t need to be defined according to their parameters. At each position $\mathbf{q} \in \Xi$ of environment Ξ, the coverage for a single sensor is thus the multiplication of the three above conditions. The coverage function is probabilistic as each of the membership functions provide the probability that an object of interest at point $\mathbf{q}$ is detected by the sensor $\mathbf{s}_i$. Therefore, $C(\mathbf{s}_i, \mathbf{q})$ represents the probability of coverage while $1 - C(\mathbf{s}_i, \mathbf{q})$ gives the probability of non-coverage. If more than one sensor covers $\mathbf{q}$, then a way to compute the local network coverage C_l is:

$$C_l(N, \mathbf{q}) = 1 - \prod_{i=1,\ldots,n} \left(1 - C(\mathbf{s}_i, \mathbf{q})\right). \tag{3}$$

This formulation is based on the assumption that the coverage of several sensors with respect to one position in the environment is independent from each other. This assumption roots in the probabilistic coverage definition of each sensor. Each position $\mathbf{q}$ is also attributed with another parameter $w_q \in [0, \infty)$. This parameter defines the importance of location $\mathbf{q}$ for the coverage task. Therefore, higher values of w_q represents higher importance of the location $\mathbf{q}$ in the goal coverage problem. The global coverage C_g can be:

$$C_g(N, \Xi) = \frac{1}{\sum_{\mathbf{q} \in \Xi} w_q} \sum_{\mathbf{q} \in \Xi} w_q C_l(N, \mathbf{q}). \tag{4}$$



(a) Top view (b) Side view

Figure 3: Visibility function behaviour. Assuming that the visibility between two points $\mathbf{p}$ and $\mathbf{q}$ is of question, the grey cells represent the set of cells whose elevation should be compared with the straight line between points $\mathbf{p}$ and $\mathbf{q}$.

Given an environment Ξ, the problem statement is thus to determine the sensor network deployment N that maximizes $C_g(N, \Xi)$.

2.2 Visibility Function

Visibility function calculates the visible area for each sensor. The main factor which affects the visibility between two points is the elevation of all points in the straight line connecting those two points. This information is provided by a DEM, which is basically a two dimensional matrix, where each cell stores the elevation of the corresponding location in the real environment (see Fig.3).

In order to calculate visibility between two points $\mathbf{p}$ and $\mathbf{q}$, the list of all cells in the matrix which intersects with the line-of-sight between those two points should be first calculated. Then each point in the list is checked versus the line connecting points $\mathbf{p}$ and $\mathbf{q}$. If the elevation of an intermediate point is more than the elevation of the line at that point, then points $\mathbf{p}$ and $\mathbf{q}$ are not inter-visible, otherwise they are inter-visible.

Assuming that point $\mathbf{p}$ represents the location of a sensor in the environment, first the elevation of point $\mathbf{p}$ is increased by the elevation of the sensor and then the mentioned visibility calculation process is repeated between point $\mathbf{p}$ and all other points in its vicinity. In order to save computation, this process is repeated only for points whose distance is less than the maximum coverage range threshold of the sensor, over which the coverage of the sensor is almost zero.

2.3 Probabilistic Membership Functions

The membership functions μ_d, μ_p, and μ_t can be defined as crisp function, with value of 1 when the position is within a fixed sensing range or angle of view, and otherwise 0.

$$\begin{aligned}
\mu_d(\|\mathbf{p}_i - \mathbf{q}\|) &= \begin{cases} 1 & \|\mathbf{p}_i - \mathbf{q}\| \leq d_{max} \\ 0 & \text{otherwise} \end{cases} \\
\mu_p(\angle_p(\mathbf{q} - \mathbf{p}_i) - \theta_i) &= \begin{cases} 1 & (\angle_p(\mathbf{q} - \mathbf{p}_i) - \theta_i) \in [-a, a] \\ 0 & \text{otherwise} \end{cases} \\
\mu_t(\angle_t(\mathbf{q} - \mathbf{p}_i) - \xi_i) &= \begin{cases} 1 & (\angle_t(\mathbf{q} - \mathbf{p}_i) - \xi_i) \in [-b, b] \\ 0 & \text{otherwise} \end{cases}
\end{aligned}$$

However, such functions used in a coverage function provide essentially a binary 0/1 signal, which is not a realistic performance model of real sensors. Moreover, for some local optimization methods, we would need coverage functions that are differentiable. Therefore, we propose to use probabilistic membership functions that provide a monotonically decreasing membership value over distance and relative angle of position to sensor. Hence, these functions have two benefits: first, they better comply with the performance of the real sensors, and second, they are differentiable.

We propose to use sigmoid function for distance membership function:

$$\mu_d(\|\mathbf{p}_i - \mathbf{q}\|) = 1 - \frac{1}{1 + \exp\left(-\beta_d(\|\mathbf{p}_i - \mathbf{q}\| - t_d)\right)}, \tag{5}$$

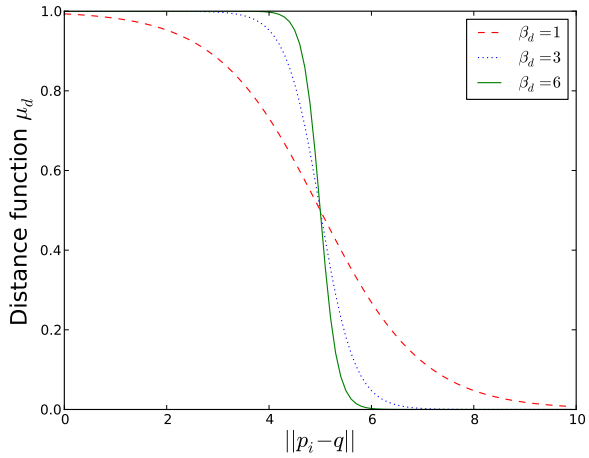


Figure 4: The effects of variations of β_d on $\mu_d(\|\mathbf{p}_i - \mathbf{q}\|)$. Assuming that $t_d = 5$.

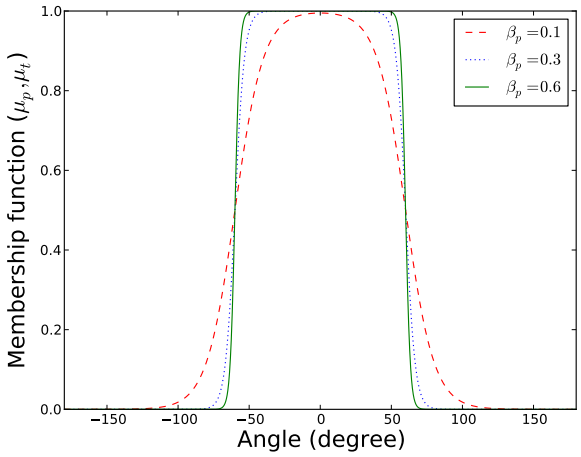


Figure 5: The effects of variations of β_p on $\mu_p(\theta_i - \angle_p(\mathbf{q} - \mathbf{p}_i))$. Assuming that $t_p = 60$.

with β_d and t_d as the parameters configuring the membership function (see Figure 4). Parameter β_d can be approximated using experimental observations on sensor behaviours. As shown in the figure, parameter β_d controls the slope of the function and t_d determines the distance where the sensor has 50% of its maximum coverage.

As for the pan angle membership functions, we propose another sigmoid function:

$$\mu_p(\underbrace{\angle_p(\mathbf{q} - \mathbf{p}_i) - \theta_i}_{\gamma_i}) = \frac{1}{1 + \exp\left(-\beta_p(\gamma_i + t_p)\right)} - \frac{1}{1 + \exp\left(-\beta_p(\gamma_i - t_p)\right)}, \tag{6}$$

where t_p controls the "width" of the function and β_p controls the slope of the function at the boundaries (see Figure 5). Note that the proposed function has the range [-180,180] degrees. Therefore, any calculated angle should be brought into this range accordingly. In the same way, membership function μ_t is defined as:

$$\mu_t(\underbrace{\angle_t(\mathbf{q} - \mathbf{p}_i) - \xi_i}_{\zeta_i}) = \frac{1}{1 + \exp\left(-\beta_t(\zeta_i + t_t)\right)} - \frac{1}{1 + \exp\left(-\beta_t(\zeta_i - t_t)\right)}, \tag{7}$$

with a range in [-90,90].

For a reasonable model of a sensor, we propose to use the parameters shown in Table 1. With these values, the sensors have 50% of the maximum coverage at 30m or at sensing angle of 120^o (see Figure 6 for an illustration of the coverage obtained).

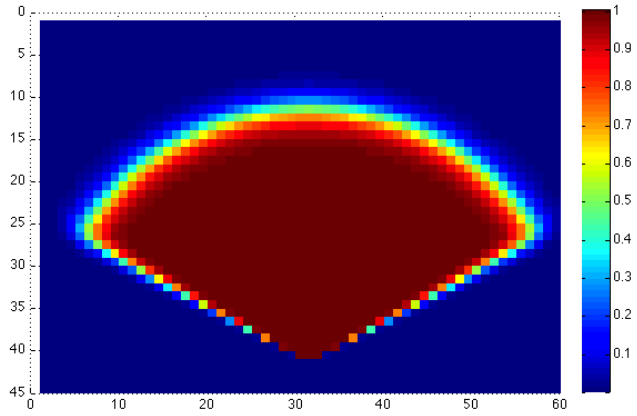


Figure 6: Probabilistic coverage model of a sensor. Assuming that a sensor is positioned at (30,40) heading upward, the colour shows different degrees of coverage for points inside the map.

Table 1: The parameter values for a realistic model of a sensor which has a 50% of the maximum coverage at 30m or span of view of 120^o.

Parameter	β_d	t_d	β_p	t_p	β_t	t_t
Value	30	1	60	1	30	1

3 Implemented Optimization Methods

From this model for sensor placement optimization, we compare four sensor placement schemes: a deterministic approach found in the WSN literature, an adaptation of Simulated Annealing (SA) [18] for sensor placement, the limited-memory Broyden-Fletcher-Goldfarb-Shanno method (L-BFGS) [7], and the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [13], an evolutionary algorithm for real-valued optimization. The deterministic approach is purely geometrical and does not take into account the model proposed in the previous section. As for the three other optimization methods, they have been applied on a real-valued vector composed of four values per sensor $(x_i, y_i, \theta_i, \xi_i)$, so as to maximize the global sensor network coverage $C_g(N, \Xi)$ over a given elevation map Ξ:

$$N = \{(x_1, y_1, \theta_1, \xi_1), (x_2, y_2, \theta_2, , \xi_2), \ldots, (x_n, y_n, \theta_n, \xi_n)\},$$
$$N^* = \text{argmax}_N C_g(N, \Xi).$$

We briefly explain each method in the following sections.

3.1 Deterministic Sensor Placement

The deterministic method has been shown to achieve full coverage on the Cartesian plane [5, 14]. Figure 7 illustrates this placement pattern, where sensors are organized in layers of horizontal stripes. Assuming sensors with sensing range r_s, they are simply distributed $\sqrt{3}\,r_s$ apart on every stripe, and the stripes are themselves separated from one another by $\frac{3}{2}r_s$. Furthermore, the stripes are interleaved to form a triangular lattice pattern. This approach does not take the terrain into consideration.

As presented in Figure 6, the pan angle coverage of each sensor is roughly 120^o. Therefore, to obtain an omnidirectional coverage at each position (because the deterministic approach assumes that all sensors have omnidirectional coverage), we place three sensors facing 120^o degrees apart from each other, at each position specified by the deterministic method.

3.2 Simulated Annealing Method on Single Sensor

SA [18] is a stochastic optimization algorithm. With a generic probabilistic heuristic approach, simulated annealing may escape local optimum and converge to global optimum, and thus may be more effective for a global optimization problem of a given function in a large search space. Our implementation of SA is described in Figure 9. It requires the definition of the temperature function (temperature(t)), and the setting of two parameters (M, σ)

- Parameter M defines the maximum iterations for simulated annealing. The larger the M, the more time consuming the optimization, and the more likely the global optimum can be reached.

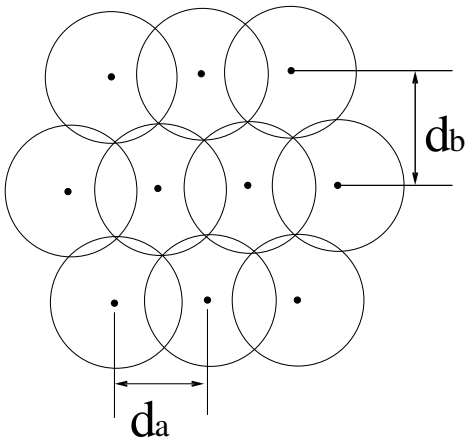


Figure 7: Pattern of the deterministic method [5, 14] implemented in the paper, where $d_a = \sqrt{3}r_s$, $d_b = \frac{3}{2}r_s$, and r_s is sensing range for a sensor. Circles are sensor sensing ranges, and dots are sensor positions.

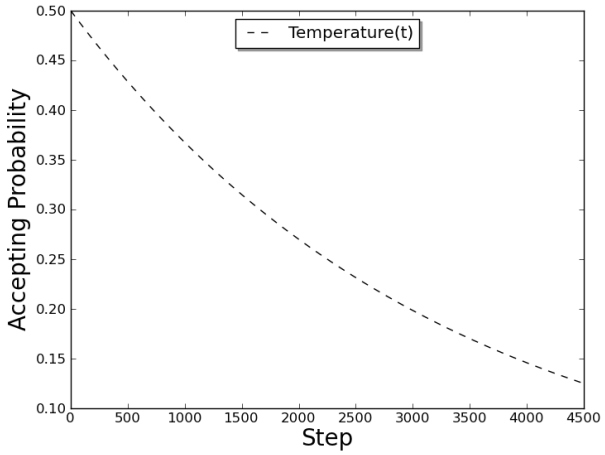


Figure 8: Illustration of the temperature function used for the simulated annealing method. Here, it is assumed that the maximum number of iterations (M) is 4550.

- σ defines candidate generator, i.e., the size of neighbourhood where subsequent solutions will be generated. An essential requirement for σ is that it must provide a sufficiently short path from the initial state to any state which may be the global optimum. Another issue is that σ should be selected so that the search path avoids becoming trapped in a local minimum, i.e., it must be large enough to cross local minima in an effort to reach the global optimum.
- Temperature function (temperature(t)) defines the probability of accepting a move in simulated annealing. Initially, the temperature(t) is set to a high value, then it is decreased at each step according to some annealing schedule, and finally ends with temperature(t) $\rightarrow 0$ towards the end of the allocated maximum steps, M. When the temperature is high, the probability of accepting a move will be high. When the cooling rate is low, the probability of accepting a move decreases. The idea is that the system is expected to wander initially towards a broad region of the search space containing good solutions, ignoring small features of the energy function; then drift towards low-energy regions that become narrower and narrower. To satisfy the conditions above, the temperature(t) is defined as an exponential decay function (see Figure 8) as follows:

$$\text{temperature}(t) = \frac{1}{2}\exp\left[-\frac{2\ln 2 \times t}{M}\right] \tag{8}$$

3.3 Limited-memory BFGS method

BFGS [24] is a numerical optimization method for solving non-linear optimization problems. This method is an example of Quasi-Newton optimization methods, which find the stationary point of a function without computing the Hessian

Initialize sensor network with random positions uniformly distributed in placement domain (assuming domain to be in $\mathbf{p}_i \in [0,1]^2$), and random orientations,

$$\begin{aligned} x_i &\sim \text{Unif}(0,1),\ i=1,\ldots,n, \\ y_i &\sim \text{Unif}(0,1),\ i=1,\ldots,n, \\ \theta_i &\sim \text{Unif}(0,1),\ i=1,\ldots,n, \\ \xi_i &\sim \text{Unif}(0,1),\ i=1,\ldots,n, \\ N &= \{(\mathbf{p}_1,\theta_1,\xi_1),(\mathbf{p}_2,\theta_2,\xi_2),\ldots,(\mathbf{p}_n,\theta_n,\xi_n)\}. \end{aligned}$$

Assess performance of initial sensor network, $f = C_g(N,\Xi)$.
Set best sensor network and best performance to the initial one, $N_{\text{best}} = N,\ f_{\text{best}} = f$.
for $t = 1,\ldots,M$ **do**
  Select a random sensor with uniform distribution, $s \sim \text{DiscUnif}(n)$.
  Apply a random perturbation of sensor position $\mathbf{p}_s = (x_s, y_s)$ and orientation (θ_s, ξ_s) to generate new candidate sensor network placement N',

$$r_x \sim \mathcal{N}(0,\sigma),\ r_y \sim \mathcal{N}(0,\sigma),\ r_\theta \sim \mathcal{N}(0,\sigma),\ r_\xi \sim \mathcal{N}(0,\sigma),$$
$$x'_s = x_s + r_x,\ y'_s = y_s + r_y,\ \theta'_s = \theta_s + r_\theta, \xi'_s = \xi_s + r_\xi,$$
$$N' = \{(\mathbf{p}_1,\theta_1,\xi_1),\ldots,(\mathbf{p}'_s,\theta'_s,\xi'_s),\ldots,(\mathbf{p}_n,\theta_n,\xi_n)\}.$$

  Evaluate performance of new candidate sensor network N', $f' = C_g(N',\Xi)$.
  if $f' > f$, the new sensor network is better than current one, **then**
    Accept new sensor network, $N = N',\ f = f'$.
    if $f' > f_{\text{best}}$, new sensor network is better than best so far, **then**
      Set best sensor network and best performance to the current one, $N_{\text{best}} = N',\ f_{\text{best}} = f'$.
    end if
  else
    Get temperature of current iteration, $E_i = \text{temperature}(t)$.
    if $e_i < E_i,\ e_i \sim \text{Unif}(0,1)$, current sensor network is accepted given the temperature, **then**
      Accept new sensor network, $N = N',\ f = f'$.
    end if
  end if
end for
Return best sensor network found, N_{best}, as final result.

Figure 9: Pseudo-code of sensor placement with simulated annealing, with perturbation of one sensor position at a time.

matrix of the objective function. It rather updates an estimate of the inverse Hessian matrix. Limited-memory BFGS (L-BFGS) stores few past inverse Hessian matrix updates instead of the full matrix. The method can make an estimation of the derivatives through numerical computation, or use the analytical formula. With the numerical gradients which are used here, L-BFGS can be considered as a black-box deterministic optimization method. Moreover, compared to other black-box methods, it scales better for problems with a high number of variables.

The execution of L-BFGS algorithm needs determination of four parameters, namely, m, ϵ, f, and p. Parameter m defines the maximum number of past updates over the approximation of the Hessian Matrix stored in the algorithm. Parameter ϵ represents the step size used for the numerical calculation of the objective's function gradient. Parameters f and p define the stopping criteria, so that, the iteration of the algorithm stops if any of the following conditions become true [32]:

$$\frac{(x_k - x_{k+1})}{\max(|x_k|,|x_{k+1}|,1)} \leq f * ep, \tag{9}$$

$$\max|pg_i| \leq p \tag{10}$$

where ep is the machine precision, and pg_i is the i^{th} component of the gradient projection. Equations 9 and 10 make sure that the algorithm will stop when the size of the correction updates over the Hessian matrix becomes very small.

3.4 CMA-ES Evolutionary Algorithm

CMA-ES [13] is an evolutionary algorithm known for its good performance and stability [11]. It updates the covariance matrix of the distribution to learn a second order model of the underlying objective function, similar to the approximation of the inverse Hessian matrix in the Quasi-Newton method in classical optimization. However, it does not require analytical derivatives of the partial derivatives.

For sensor placement optimization, the position and orientation of the sensors can be encoded inside an individual, and a population of individuals can be evolved through generations. At the end of the evolution, the individual with the best coverage is chosen as the final solution.

The algorithm's parameters include the number of parents (μ), the number of offspring (λ), the mutation factor (σ), and the number of generations through which the algorithm runs. In each generation of the algorithm, a collection of the best μ candidate solutions are selected from the set of λ offspring of the previous generation. These solutions are then used to update the distribution parameters, which will eventually generate the offspring for the next generation.

Table 2: Information about the test maps used for the experiments.

Method	NC-A	NC-B	NC-C	NC-D	NC-E	ULaval
West boundary	638775	638684	638606	638300	638820	245615
East boundary	638820	638774	6386686	638480	639000	245915
South boundary	220250	220250	220170	220615	220220	5182550
North boundary	220295	220295	220250	220750	220490	5182850
Highest elevation	115.9	116.4	120.3	131.5	123.8	140.1
Lowest elevation	112.7	112.6	112.	123.9	111.5	80.7
No. of Columns	45	90	80	180	180	300
No. of Rows	45	45	80	135	270	300
No. of grid points	2025	4050	6400	24300	48600	90000

3.5 Experiments

To conduct our experiments, we selected a mountainous area in North Carolina. The data was provided by a raster layer map in the "OSGeo Edu" dataset, that stores geospatial information about parts of the North Carolina State, USA. More specifically, we focus on a portion of the map that covers a small watershed in a rural area near NC capital city, Raleigh. The coordinate system of the map is the NC State Plane (Lambert Conformal Conic projection), metric units, and NAD83 geodetic datum. We used five portions of the map for our experiments (see Figure 10). The information about different selected portions of the map is presented in Table 2. Testing the optimization methods with different map sizes, allows to check the scalability of each method over different map sizes.

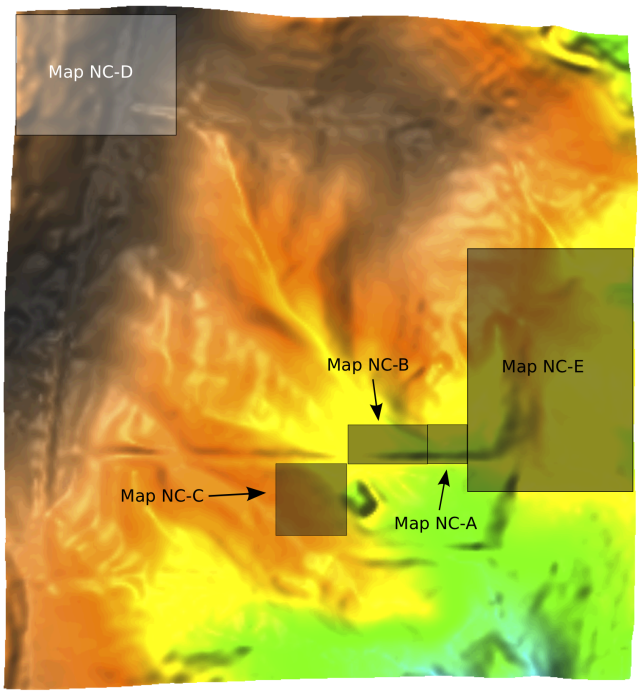


Figure 10: Map of the watershed area chosen for the experiments. The test areas have been highlighted inside the map.

We also tested the optimization algorithms over a map of the campus of Université Laval. The map of the area is shown in Figure 11. In this experiment, which is an example of a surveillance system for the campus, the goal is twofold. First we want to test the performance of each algorithm in the presence of man-made obstacles (i.e. buildings). Second, the target area is weighted, meaning that each pixel is attributed with a different weight ($w_{\mathbf{q}}$) as described in Sec. 2. For this experiment, we assume that the top of the buildings have no importance in the total coverage (i.e. $w_{\mathbf{q}} = 0$), the streets have an average importance (i.e. $w_{\mathbf{q}} = 0.4$), and the ground level where the pedestrians walk have the highest importance (i.e. $w_{\mathbf{q}} = 0.8$).

Sensors are modelled following a description given in Sec. 2, using the parameters presented in Table 1. For all methods except for the deterministic approach, each sensor placement optimization scheme was run 30 times, from which

Available at `http://grass.itc.it/download/data.php`.

Figure 11: Part of the Université Laval map, chosen for the weighted experiments. Here, different parts of the map have different weights. Buildings are shown in red and have the weight $w_{\mathbf{q}} = 0$, ground in represented in green and have the weight $w_{\mathbf{q}} = 0.8$, and the streets are shown in black and have the weight $w_{\mathbf{q}} = 0.4$.

Table 3: The value of the ϵ parameter for the L-BFGS method. Each dimension of freedom has a different value calculated by $\epsilon_r = \frac{1.}{R_r}$, where ϵ_r and R_r are the ϵ and range for dimension r respectively.

Parameter	NC-A	NC-B	NC-C	NC-D	NC-E	ULaval
ϵ_x	0.023	0.011	0.012	0.0056	0.0056	0.0033
ϵ_y	0.023	0.023	0.012	0.0074	0.0037	0.0033
ϵ_θ	0.0028	0.0028	0.0028	0.0028	0.0028	0.0027
ϵ_ξ	0.0056	0.0056	0.0056	0.0056	0.0056	0.0056

are estimated the average and the standard deviation of each method. The initial position and orientation of the sensors were determined randomly for each run of each method. CPU times are also averaged over the 30 runs, in order to compare the resources required by each method to produce a solution. These time values have been evaluated by running the methods on one core of Intel i7 computers clocked at 2.8 GHz.

For simulated annealing the perturbations for positions and for orientation is a Gaussian distribution with standard deviation σ_{sa}. The optimal value for σ_{sa} is found by trial and error, and set to $\sigma_{sa} = 0.01$ for each map. With L-BFGS, a history of the $m = 20$ past updates of the position and gradient are used to limit the memory usage, and the stop criteria is parametrized by values $f = 10$ and $p = 1.0^{-10}$. The other parameter for L-BFGS is the ϵ used for numerical calculation of the derivative. In our implementation, all the parameter values are assumed to be in $[0, 1]$ boundary, therefore, each map would have a different ϵ value with respect to each of its free parameters. These values are reported in Table 3 CMA-ES is run with a population of $\lambda = \lfloor 4 + 3 * \log(N) \rfloor$ offspring and $\mu = \lfloor \frac{\lambda}{2} \rfloor$ parents, for 350 generations. Here, N is the dimensionality of the given problem, determined by the number of sensors in each map. A mutation factor $\sigma = 0.167$ is also used. All the parameters are summarized in Table 4.

The stop criterion for L-BFGS method has been explained in Section 3 and depends on the value of parameters f, p, and ϵ. For the stop criterion of CMA-ES optimization method, we run the method until convergence. To check for convergence, we run the method and report the best solution found for each iteration. If the best solution found does not improve (meaning that the difference is below 0.1%) for a specific number of iterations (referred to as τ), then we assume that the algorithm has converged and we stop the algorithm. The value of τ for each map depends on the size of the map and the number of sensors needed to cover the map. It is an estimate for the complexity of the map. The bigger the map, the bigger the value of τ, because the speed of convergence decreases as the size of the map increases. The τ value for each map is reported in Table 5.

Table 4: The parameter values for simulated annealing, L-BFGS, and CMA-ES optimization methods.

	SA	L-BFGS			CMA-ES
Parameter	σ_{sa}	m	f	p	σ
Value	0.01	20	10	1.0^{-10}	0.167

Table 5: The number of iterations (τ) for which CMA-ES method is checked for convergence. It means that if the performance of a method does not improve during the mentioned number of iterations, it is assumed that the method has converged and the algorithm is stopped.

Map	NC-A	NC-B	NC-C	NC-D	NC-E	ULaval
τ	50	100	150	225	300	400

Table 6: The average number of iterations for the CMA-ES method, which is used to calculate the maximum number of iterations for the SA method. Here, the maximum number of iterations is calculated by multiplying the average number of iterations by the number of function evaluations in each iteration λ.

Map	NC-A	NC-B	NC-C	NC-D	NC-E	ULaval
Average iterations, CMA-ES	171	256	338	700	1037	2944
λ, CMA-ES	11	13	14	18	20	22
Maximum iterations, SA	1881	3328	4732	12600	20740	64768

We cannot take the same approach for the SA method. The reason lies in the definition of the temperature function, as explained in Section 3. The temperature function needs the value of the maximum number of iteration as an input parameter. For this purpose, we run the CMA-ES method with the mentioned setting, and for each map, take the average number of iterations needed for the method to converge. Next, the number of passed iterations is multiplied by the number of function evaluations at each iteration to reach the total number of function evaluations that the CMA-ES method has performed on each map. This value is assigned as the maximum number of iterations for the SA method, because SA performs only one function evaluation in each iteration. In Table 6 we have reported the average number of iterations for the CMA-ES value and the maximum number of iterations calculated for the SA method.

All optimization programs are written in Python, except the line-of-sight calculation which was implemented in C++ to gain computation speed. We used the implementation of L-BFGS from the well-known SciPy library. CMA-ES implementation was taken from DEAP, a Python library for evolutionary algorithms developed at Université Laval.

3.6 Experimental Results

We compare the performance of the four mentioned placement methods, that is Deterministic approach, Simulated Annealing, L-BFGS, and CMA-ES.

We ran each optimization scheme 30 times and calculated the corresponding coverage percentage on the target areas. A qualified sensor optimization scheme should have high coverage and low standard deviations of coverage given a number of runs. In other words, we are evaluating each algorithm in terms of both accuracy and robustness. The results of each method using the best parameter sets are shown in the Table 7. In order to show the statistical significance of the results, we also performed a Student's t-test on the two methods having the best results. We would reject the hypothesis that there is a significant difference between the performance of those two methods if the resulting p-value is above 0.05. We have also presented the optimal results found over the six mentioned maps in Figures 12 and 13.

Among tested methods, CMA-ES outperformed the other two in almost all maps, except one (NC-B), where SA produced the best result. On the smaller maps (NC-A, NC-B, and NC-C), SA produced results very close to CMA-ES in terms of coverage, but as the size of the maps increase, the difference between the coverage in the two mentioned methods become more apparent. In larger maps the performance of SA becomes closer to L-BFGS, which performs worse than the other two methods in general. In terms of the standard deviation, SA produced more stable results compared to the two other methods.

L-BFGS performs a local search, therefore, it is not surprising that it performed worse than the other two global optimization methods. In comparison, the deterministic method produced the worst result among all others, as it does not take terrain into consideration.

With respect to computational power, SA and CMA-ES consumed roughly the same amount of computation on smaller maps. The reason is that on smaller maps, the main computational demand of the algorithms, lies in evaluating the coverage for the candidate solutions. But as the size of the map increases, CMA-ES requires more expensive calculations to estimate its covariance matrix.This generates the larger difference obtained on the computational requirements for higher dimensions.

The computational demand of L-BFGS also increases with the dimensionality of the search space. The main reason for high computational demand of the L-BFGS method is the numerical evaluation of the derivatives. Indeed, L-BFGS method

Available at `http://www.scipy.org`.

Available at `http://deap.googlecode.com`.

Table 7: Coverage percentage on the target areas with various number of sensors. Each scheme has been run 30 times, with coverage loss averages and the corresponding standard deviations reported. Note that 100% coverage is not possible with less sensors than the number of grid points in the map, given the probabilistic model used.

Method	NC-A	NC-B	NC-C	NC-D	NC-E	ULaval
Number of sensors	3	6	9	34	72	108
Search dimensions	12	24	36	144	288	432
Deterministic	94.4%	76.73%	70.98%	79.29%	64.09%	52.89%
SA average	95.69%	**94.80%**	**97.10%**	96.11%	96.20%	92.68%
SA stdev	0.3%	0.7%	0.6%	0.2%	0.3%	0.7%
SA CPU time	16 sec.	171 sec.	569 sec.	9203 sec.	36392 sec.	154437 sec.
CMA-ES average	**96.39%**	**94.69%**	**97.55%**	**97.90%**	**97.88%**	**96.77%**
CMA-ES stdev	0.8%	1.9%	0.6%	0.3%	0.6%	0.4%
CMA-ES CPU time	34 sec.	181 sec.	572 sec.	9730 sec.	38988 sec.	162706 sec.
L-BFGS average	91.65%	86.12%	96.10%	96.05%	96.02%	85.72%
L-BFGS stdev	4.7%	5.2%	0.7%	0.4%	0.6%	2.1%
L-BFGS CPU time	46 sec.	266 sec.	3180 sec.	41456 sec.	67566 sec.	260124 sec.

Table 8: Performance of the L-BFGS method using $f = 1e12$. Compared to Table 7, it is clear that the performance of the algorithm has decreased, but the computation time of the algorithm has also decreased significantly.

Method	NC-A	NC-B	NC-C	NC-D	NC-E	ULaval
L-BFGS average	90.52%	82.6%	95.23%	93.87%	95.46%	83.57%
L-BFGS stdev	5.6%	2.9%	1.5%	1.6%	0.8%	5.6%
L-BFGS CPU time	12 sec.	64 sec.	191 sec.	3672 sec.	15952 sec.	69122 sec.

needs to calculate derivative with respect to all the free dimensions of the problem, and for each derivative one coverage calculation needs to be done. High computational demand of L-BFGS is also related to our setting of the algorithm. As mentioned before in Section 3 and Table 4, we chose the value of the parameter f (which determines if the algorithm has converged or not) to be equal to 10. Higher values for this parameter results in much faster convergence of the algorithm, at the price of slight decrease in the performance. For example, Table 8 presents the performance of L-BFGS for $f = 1e12$. The high processing time of the method in the high accuracy setting is related to the line-search step of the algorithm. In this step the position of the next optimal point in the direction of the gradient is calculated. In the starting iterations, the algorithm can make large steps in the direction of the optimal position, but as the algorithm converges the gradient becomes less informative and the line-search mechanism needs to be restarted more often. Therefore, if the performance is not of great importance, this gradient-based methods can provide a good yet suboptimal results in a time less than the other stochastic optimization methods.

The best result for placement in Campus map is shown in Figure 13. Although the obtained result is comparatively good, there are some sensors that are covering the areas with coverage weight of zero ($w_{\mathbf{q}} = 0$). A potential extension to our current approach is to modify the optimization methods, so that in each iteration, sensors placed in areas with zero weight will be moved to the closest location with a non-zero weight value.

4 Conclusion

This paper proposes a novel model for optimization of sensor placement. The novelty of this model lies in the integration of terrain information (elevation maps) with a probabilistic sensor model. Results are reported for different optimization methods tested with this model. CMA-ES optimization method outperformed the three others (deterministic, SA, and L-BFGS). This demonstrates that the optimization problem as defined in the current framework is quite a difficult one, requiring stochastic search method.

From a modelling perspective, refinements are possible, for example by simulating signal propagation. However, our objective here is to make a proof of concept of sensor placement through the use of black-box optimization, using a probabilistic model of sensors operating in a given environment. If one has better models, the proposed optimization approaches used should still be applicable.

This serves as a starting point to further investigate the use of evolutionary algorithms in sensor placement optimization. We are considering as another future work to make use of evolutionary multi-objective optimization for sensor deployment, optimizing over multiple criteria simultaneously such as number of sensors used, energy saving, and multiple coverage.

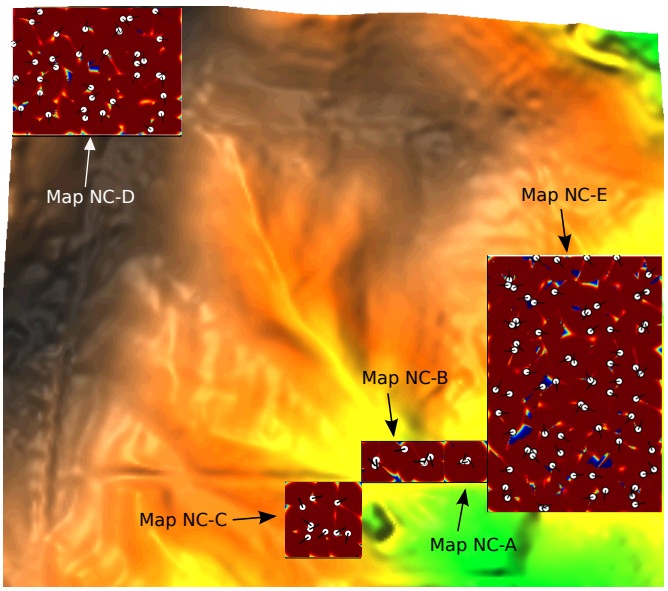


Figure 12: The optimal result obtained for each of the NC maps. Different colour represents different degree of coverage, using the same colour-map than in Fig. 6. The sensors are presented by white circles in the environment, and the black line connected to each sensor represents the direction of coverage for each sensor. All maps are produced by CMA-ES, except map NC-B which is produced by SA.

Acknowledgements

This project has been funded by the GEOIDE Network of Centres of Excellence (Canada), Defence Research and Development Canada (DRDC), and MDA Systems Ltd. We thank Patrick Maupin and Anne-Laure Jousselme from DRDC Valcartier for scientific discussions on the project. We also thank Albert Hung-Ren Ko for his early participation in the project and Annette Schwerdtfeger for proofreading the manuscript.

References

[1] N. Ahmed, S. S. Kanhere, and S. Jha. Probabilistic coverage in wireless sensor networks. *LCN '05: Proceedings of the The IEEE Conference on Local Computer Networks*, pages 672–681, 2005.

[2] V. Akbarzadeh, C. Gagné, M. Parizeau, and M.A. Mostafavi. Black-box Optimization of Sensor Placement with Elevation Maps and Probabilistic Sensing Models. *International Symposium on Robotic and Sensors Environment, ROSE*, pages 89–94, 2011.

[3] V. Akbarzadeh, A. Ko, C. Gagné, and M. Parizeau. Topography-Aware Sensor Deployment Optimization with CMA-ES. *Parallel Problem Solving from Nature (PPSN XI)*, pages 141–150, 2010.

[4] A.S. Aspbury and R.M. Gibson. Long-range visibility of greater sage grouse leks: a GIS-based analysis. *Animal Behaviour*, 67(6):1127–1132, 2004.

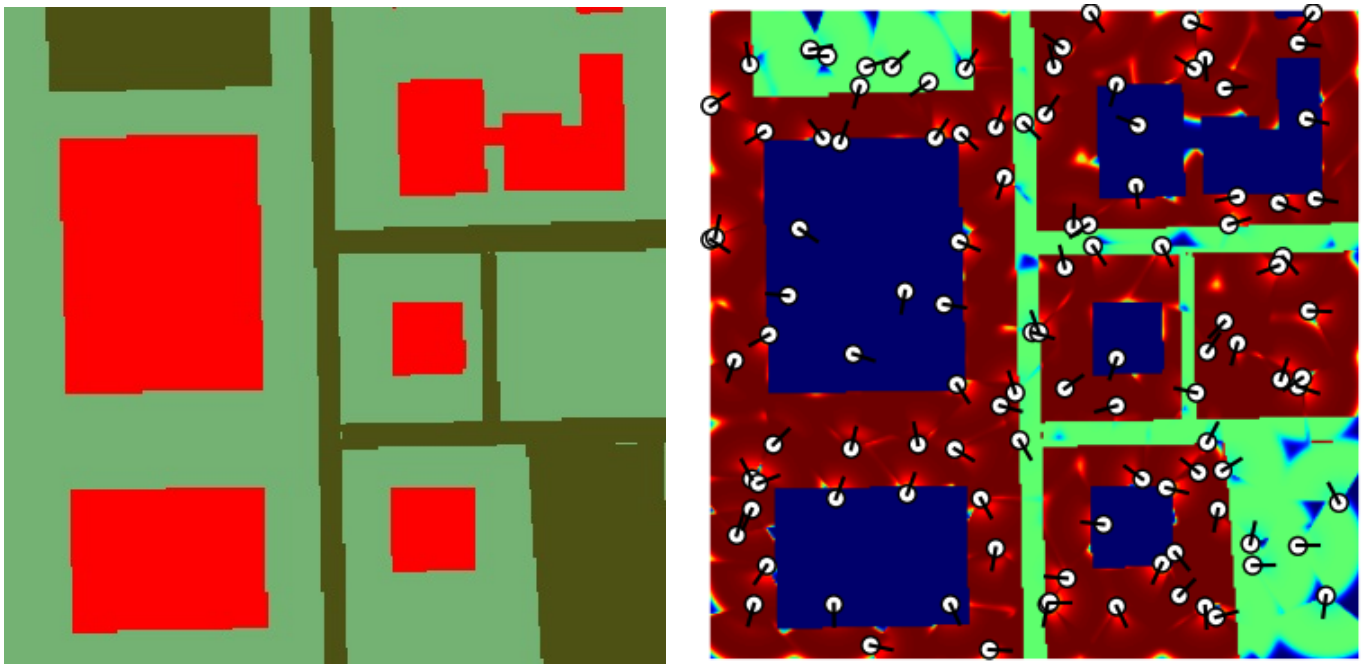
[5] X. Bai, S. Kumar, D. Xuan, Z. Yun, and T.H. Lai. Deploying wireless sensors to achieve both coverage and connectivity. In *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, pages 131–142, 2006.

[6] A. Boukerche and X. Fei. A coverage-preserving scheme for wireless sensor network with irregular sensing range. *Ad Hoc Network*, 5(8):1303–1316, 2007.

[7] R.H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[8] L. De Floriani, P. Marzano, and E. Puppo. Line-of-sight communication on terrain models. *International Journal of Geographical Information Science*, 8(4):329–342, 1994.

[9] S.S. Dhillon and K. Chakrabarty. Sensor placement for effective coverage and surveillance in distributed sensor networks. *Wireless Communications and Networking, WCNC*, 3, 2003.



(a) Map A (b) Map B

Figure 13: Result of placement on the Campus ULaval map: (a) A two dimensional view of the map with buildings in red, (weight $w_{\mathbf{q}} = 0$), streets in dark green (weight $w_{\mathbf{q}} = 0.4$) and the ground shown in light green (weight $w_{\mathbf{q}} = 0.8$). (b) Optimal result obtained with CMA-ES, having a coverage rate of 97.3%. Here sensors are shown as white circles, with the black lines representing the coverage direction. Different colors in (b) shows different coverage values using the same colour-map than in Fig.6

[10] W.R. Franklin and C. Vogt. Multiple observer siting on terrain with intervisibility or lo-res data. In *XXth Congress, International Society for Photogrammetry and Remote Sensing, Istanbul*, pages 12–23, 2004.

[11] S. Garcia, Daniel Molina, Manuel Lozano, and Francisco Herrera. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization. *Journal of Heuristics*, 6(15):617–644, 2009.

[12] M.A. Guvensan and A.G. Yavuz. On coverage issues in directional sensor networks: A survey. *Ad Hoc Networks*, 2011.

[13] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159 – 195, 2001.

[14] M. Hefeeda and H. Ahmadi. Energy efficient protocol for deterministic and probabilistic coverage in sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 99:579–593, 2009.

[15] M.M. Holland, R.G. Aures, and W.B. Heinzelman. Experimental investigation of radio performance in wireless sensor networks. In *Wireless Mesh Networks, 2006. WiMesh 2006. 2nd IEEE Workshop on*, pages 140–150. IEEE, 2006.

[16] S. Banerjee K. Kar. Node placement for connected coverage in sensor networks. *Proceedings of the Workshop on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2003.

[17] D. Kidner, A. Sparkes, and M. Dorey. GIS and wind farm planning. *Geographical Information and Planning*, pages 203–223, 1999.

[18] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *science*, 220(4598):671, 1983.

[19] Benyuan Liu and Don Towsley. A study of the coverage of large-scale sensor networks. In *Proc. of the IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS04)*, pages 475–483, 2004.

[20] P. Lv, J. Zhang, and M. Lu. An optimal method for multiple observers sitting on terrain based on improved simulated annealing techniques. *Advances in Applied Artificial Intelligence*, pages 373–382, 2006.

[21] H. Ma, X. Zhang, and A. Ming. A coverage-enhancing method for 3d directional sensor networks. In *IEEE International Conference on Computer Communications (INFOCOM)*, pages 2791–2795. IEEE, 2009.

[22] A.T. Murray, K. Kim, J.W. Davis, R. Machiraju, and R. Parent. Coverage optimization to support security monitoring. *Computers, Environment and Urban Systems*, 31(2):133–147, 2007.

[23] J. V. Nickerson and S. Olariu. Protecting with sensor networks: Attention and response. *Proceedings of the 40th Annual Hawaii International Conference on System Sciences*, 2007.

[24] J. Nocedal and S.J. Wright. *Numerical optimization*. Springer verlag, 1999.

[25] S. Olariu and J.V. Nickerson. Protecting with sensor networks: perimeters and axes. In *Military Communications Conference, 2005. MILCOM 2005. IEEE*, pages 1780–1786. IEEE, 2005.

[26] J. Balogh S. Kumar, T. H. Lai. On k-coverage in a mostly sleeping sensor network. *Wireless Network*, 14:277 – 294, 2006.

[27] H.R. Topcuoglu, M. Ermis, and M. Sifyan. Positioning and utilizing sensors on a 3-d terrain part i—theory and modeling. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, (99):1–7, 2010.

[28] Y. C. Wang and Y. C. Tseng. Distributed deployment schemes for mobile wireless sensor networks to ensure multilevel coverage. *IEEE Transactions on Parallel and Distributed System*, 19(9):1280 – 1294, 2008.

[29] M. Worboys and M. Duckham. *GIS: A computing perspective*. CRC, 2004.

[30] J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. *Computer Networks*, 52(12):2292–2330, 2008.

[31] H. Gupta Z. Zhou, S. Das. Connected k-coverage problem in sensor network. *Proceedings of 13th International Conference on Computer*, 1, 2007.

[32] C. Zhu, R.H. Byrd, and J. Nocedal. L-BFGS-B: Algorithm 778: L-BFGS-B, fortran routines for large scale bound constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, 1997.

[33] Y. Zou and K. Chakrabarty. A distributed coverage- and connectivity-centric technique for selecting active nodes in wireless sensor networks. *IEEE Transaction on Computers*, 54(8):978–991, 2005.