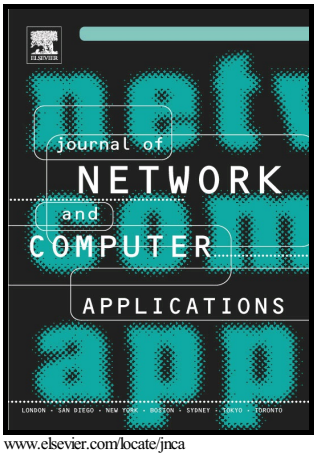
On Balanced *k*-Coverage in Visual Sensor Networks

Sakib Md. Bin Malek, Md. Muntakim Sadik, Ashikur Rahman



www.elsevier.com/locate/jnca

PII: S1084-8045(16)30140-0
DOI: http://dx.doi.org/10.1016/j.jnca.2016.06.011
Reference: YJNCA1668

To appear in: *Journal of Network and Computer Applications*

Received date: 12 December 2015
Revised date: 1 June 2016
Accepted date: 24 June 2016

Cite this article as: Sakib Md. Bin Malek, Md. Muntakim Sadik and Ashiku Rahman, On Balanced *k*-Coverage in Visual Sensor Networks, *Journal o Network and Computer Applications* http://dx.doi.org/10.1016/j.jnca.2016.06.011

On Balanced k-Coverage in Visual Sensor Networks

Sakib Md. Bin Malek[a], Md. Muntakim Sadik[a], Ashikur Rahman[a]

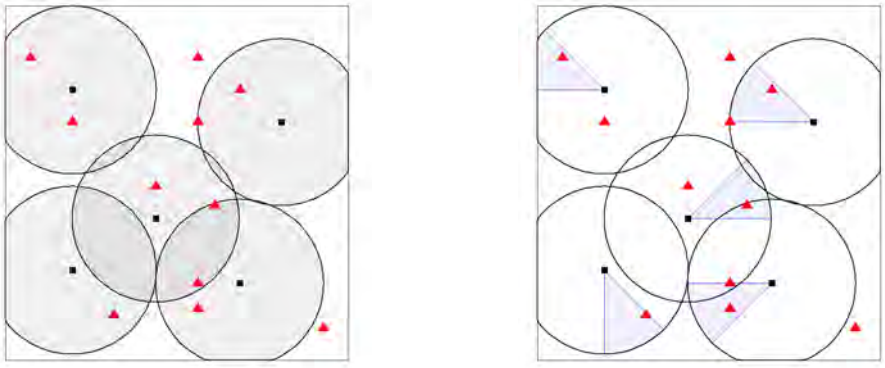
[a]*Department of Computer Science and Engineering*
Bangladesh University of Engineering and Technology, Dhaka, Bangladesh

Abstract

Given a set of directional visual sensors, the k-coverage problem determines the orientation of minimal directional sensors so that each target is covered at least k times. As the problem is NP-complete, a number of heuristics have been devised to tackle the issue. However, the existing heuristics provide imbalance coverage of the targets–some targets are covered k times while others are left totally uncovered or singly covered. The coverage imbalance is more serious in under-provisioned networks where there do not exist enough sensors to cover all the targets k times. Therefore, we address the problem of covering each target at least k times in a balanced way using minimum number of sensors. We study the existing Integer Linear Programming (ILP) formulation for single coverage and extend the idea for k-coverage. However, the extension does not balance the coverage of the targets. We further propose Integer Quadratic Programming (IQP) and Integer Non-Linear Programming (INLP) formulations that are capable of addressing the coverage balancing. As the proposed formulations are computationally expensive, we devise a faster Centralized Greedy k-Coverage Algorithm (CGkCA) to approximate the formulations. Finally, through rigorous simulation experiments we show the efficacy of the proposed formulations and the CGkCA.

Keywords: VSN, k-coverage, Balanced coverage, ILP, IQP, INLP, Greedy Algorithm

Email addresses: 0905039.smbm@ugrad.cse.buet.ac.bd (Sakib Md. Bin Malek), 0905003.mms@ugrad.cse.buet.ac.bd (Md. Muntakim Sadik), ashikur@cse.buet.ac.bd (Ashikur Rahman)



(a) Omnidirectional Sensors (b) Directional Sensors

Figure 1: Omnidirectional and Directional Sensors

1. Introduction

A visual sensor network (VSN), also known as a Smart Camera Network (SCN), consists of a set of targets to be monitored by a set of *smart* (visual) sensors capable of self-controlling their orientations and ranges. Such VSNs have drawn considerable attention of researchers due to their enormous applicability in real-world scenarios like surveillance system, environment monitoring, smart traffic controlling system etc., to name a few.

1.1. Coverage in Visual Sensor Network

The way the targets are covered or monitored by the visual sensors or cameras in an VSN is very important and the complexity of the coverage depends on whether omnidirectional or directional sensors are used. Fig. 1 shows how omnidirectional (360°) and directional sensors work. A target is said to be *covered* by a sensor if that target lies within a active sensor's sensing range and that active sensor is oriented towards that target. Fig. 2 will clarify the term *coverage*. Fig. 2 shows a sensor (black rectangle) and 4 targets (red triangles). Target 3 is not within the sensing range of the sensor, thus can not be covered

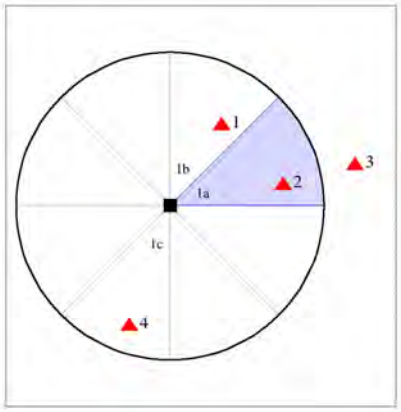


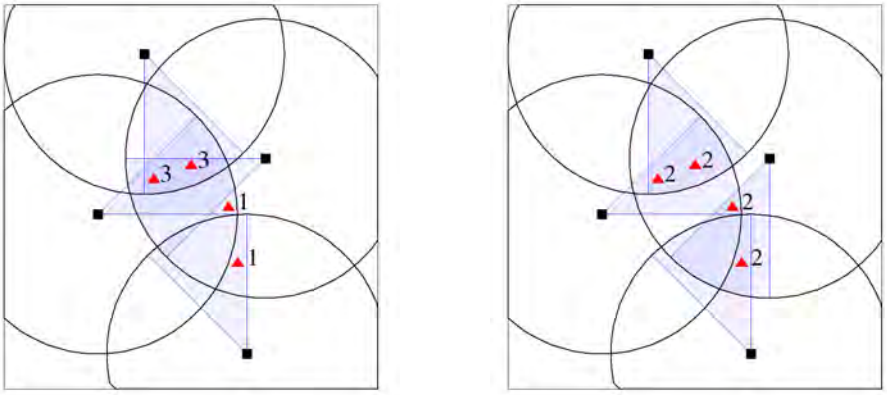
Figure 2: Sensor covering targets

by it. If the sensor is a omnidirectional targets 1, 2, and 4 will be covered. If the sensor is directional and is oriented towards pan 1a, as portrayed in Fig. 2, then only target 2 will be covered. If it was oriented to pan 1b and 1c, then targets 1 and 4 will be covered respectively.

1.2. Fault Tolerance in Visual Sensor Network

The primary goal of VSNs is to monitor as many targets as possible [1], [2]. However, if the sensor covering a target malfunctions, runs out of power, or if the line of sight is blocked by a perpetrator, a previously covered target may suddenly become uncovered. The simplest solution to this problem is to incorporate *fault tolerance* besides *coverage*, i.e., to cover the target by more than one sensor. This joint *fault tolerant coverage* problem is well-known as "*k-coverage*" problem in the literature.

Formally, our work tackles the k-coverage problem, where each target is to be covered by at least k sensors ($k \geq 1$). The efficiency of the solution to the problem depends on the extent of camera usage as fewer number of active sensors implies lower energy consumption and longer network life time. Thus, in k-coverage problem one needs to minimize camera-usage besides covering each target in a fault-tolerant way.



(a) Imbalanced coverage (b) Balanced coverage

Figure 3: Coverage imbalance in under-provisioned systems

1.3. Categories of Visual Sensor Network

We envision two kinds of visual sensor networks:

- under-provisioned networks
- over-provisioned networks

We call a VSN is under-provisioned if the number of sensors is insufficient to cover all the targets at least k times and over-provisioned otherwise. Consider Fig. 3 for an example of an under-provisioned network. In this figure, there are 4 cameras (rectangular ones) and 4 targets (triangular ones) and the objective is to cover each target at least 3 times (i.e., $k = 3$). Here, every camera possesses a specific number of non-overlapping pans, of which only one can be selected in a particular deployment and each pan is defined by a field of view (FoV) angle, $\theta = \frac{\pi}{4}$. Thus, a camera can pick any one of eight ($\frac{2\pi}{\theta} = \frac{2\pi}{\frac{\pi}{4}} = 8$) disjoint pans/orientations. Note that, no orientation of 4 cameras can produce a 3-coverage for this scenario, hence the network is consequently called "under-provisioned".

1.4. Coverage Imbalance Problem

In under-provisioned networks, it is impossible to provide a complete k-coverage therefore we need to provide a more *fault resilient* solution instead. For example, let's re-consider the under-provisioned network in Fig. 3a and Fig. 3b. Although both the figures show the same deployment of cameras and targets however the orientation of the cameras are different. In Fig. 3a the targets are covered by 3, 3, 1, and 1 sensors respectively, while in Fig. 3b, each of the targets are covered by 2 sensors. However, the coverage in Fig. 3b is more *fault resilient* than the coverage in Fig. 3a because in Fig. 3a two targets are covered once, while in Fig. 3b all the targets are covered twice. We say the coverage in Fig. 3b is more *balanced* than coverage in Fig. 3a.

In this paper, we focus on under-provisioned networks and provide solutions for *coverage balancing* using minimum number of sensors. The problem is eventually an instance of the classical set *multi-cover* problem whose optimization version is known to be NP-hard [3].

1.4.1. Fairness Index

In order to define *coverage balancing*, we borrow the concept of *fairness* in resource allocation systems proposed by Jain et al. [4] and modify it for our purpose (the modification is described in a later section). The *Fairness Index*, $\mathcal{FI}$, is defined as follows [4]. Suppose we have $1, 2, 3, .., m$ components in a system and x_i is the resource allocated to the i^{th} component. The fairness index of such system will be:

$$\mathcal{FI} = \frac{(\sum x_i)^2}{m \sum x_i^2} \tag{1}$$

In VSN, the sensors are the resources and the targets are the components of the system. Therefore, we can use the fairness index to judge by how much a system is balanced. Here, the number of times each target is covered is considered as the number of resources allocated to that target. Thus, the fairness index value of Fig. 3a is:

$$\mathcal{FI} = \frac{(3+3+1+1)^2}{4 \times (3^2+3^2+1^2+1^2)} = 0.8$$

and, for Fig. 3b is:

$$\mathcal{FI} = \frac{(2+2+2+2)^2}{4 \times (2^2+2^2+2^2+2^2)} = 1.0$$

Consequently, the camera orientations in Fig. 3b has more coverage balancing than the camera orientations in Fig. 3a.

1.5. Organization of the Paper

The rest of the paper is organized as following. Section 2 provides a brief literature review on the subject matter and a summary of our contributions in this paper. Section 3 introduces the description and parameters of a Visual Sensor Network and formally defines the problem that we solve in this paper. Section 3 also discusses the shortcomings of Fairness Index in capturing coverage balance and introduces a new metric Balancing Index. Section 4 shows how to formulate Integer Linear Programming (ILP) for the k-coverage problem and discusses ILP's incapability in coverage balancing. Section 5 modifies ILP to formulate Integer Quadratic Programming (IQP) and Integer Non-Linear Programming (INLP) that incorporate coverage balancing besides k-coverage. Section 6 discusses the Centralized Greedy k-Coverage Algorithm (CGkCA). Section 7 presents the simulation results and analyzes the results and finally Section 8 concludes the paper.

Throughout the paper we use the terms "camera" and "sensor" interchangeably.

2. Related Works and Our Contributions

In this section, we will review the recent research work related to our work. We will also point out our key contributions in this paper.

2.1. Related works

A large volume of research exists for the k-coverage problem where the researchers have worked only with omnidirectional sensors [5] [6] [7] [8]. Li-Hsing Yen et al. [5] have worked with n sensors with a circular range of radius r. They

formulate an exact mathematical expression for the expected area that would be k-covered. Hafeeda and Bagheri [6] provide a novel approach for solving the k-coverage problem and have modelled it as optimal hitting set problem which is NP-hard [9]. Their proposed [6] k-coverage algorithm is inspired by the approximation algorithm in [10] and they suggest both centralized randomized k-coverage and distributed randomized k-coverage algorithms. In [7], the authors, Ammari and Das, have worked with connectivity and coverage of WSN in 3D space. They prove that if there are k sensors with spherical sensing range in a Reuleaux tetrahedron then all the targets in that Reuleaux tetrahedron will be k-covered. Most of the works discussed so far assume that the locations are known. The algorithm presented by Yigal Bejerano in [11] can efficiently verify the k-coverage without any prior location information. In [12], Qiu et al. approached the k-coverage problem using k-order Voronoi diagram. They assume the node will be mobile and suggest Distributed VOronoi based Cooperation scheme (DVOC), where nodes cooperate in hole detection and also in recovery. A novel solution to k-coverage for WSNs is proposed by Alam et al. in [13]. Their solution ensures 1-coverage at deployment time and on the occurrence of any event selected sensors will increase their sensing region and provide k-coverage to cover that event.

In [14], Tian and Georganas have devised a node scheduling algorithm which would turn off redundant sensors without reducing the overall coverage of the network. This would be achieved by turning off only those sensors whose coverage region is covered by its neighbouring active sensors. In [8], Kumar et al. provide a solution to k-coverage problem where at any point of time most of the nodes are in sleep state. The authors study the lifetime of a network using different types of node distributions such as $\sqrt{n} \times \sqrt{n}$ grid model, random uniform distribution and Poisson distribution. Liu and Towsley in [15] have studied the coverage properties in large-scale sensor networks and put forward three coverage measures for–(i) the fraction of the area covered by sensors, (ii) the fraction of sensors which can be removed without reducing the covered area, and (iii) the capability of the sensor networks to detect objects moving in the

network. Huang and Tseng [2] formulated a decision problem, which will determine whether every point in the service area of the sensor networks is covered by k-sensors, for two versions of coverage problem, namely k-UC and k-NC. In k-UC, all the sensors have same range and in k-NC, the range varies. An in-detailed analysis on energy consumption in a Visual Sensor Network testbed has been carried out by Margi et al. in [16]. In [17], Lu and Yu model k-Coverage in RFID network using Plant Growth Simulation Algorithm (PGSA). They use multi-dimensional fuzzy optimization problem to evaluate the network performance. Gupta et al. address not only k-coverage of target nodes but also m-connectivity of the sensor node of a WSN in [18]. The authors assume that the sensor will be omnidirectional and are to be placed optimally and minimally in the potential positions keeping both k-coverage and m-connectivity.

There exist quite a few works using directional sensors. Jing and Abouzeid [3] formulate the coverage problem using directional sensors as Maximum Coverage using Minimum Sensors (MCMS) problem, and provide both centralized and decentralized greedy solutions. In [19], Munishwar and Abu-Ghazaleh present a new algorithm which improves the greedy approach of [3]. The optimal solution to k-coverage problem has been proved to be NP-hard by Fusco and Gupta in [20]. They also model the sensors to have a fixed viewing angle and overlapping pans.

2.2. Our Contribution

Although many researchers [6],[11], [8], [7], [5], [17], [18], [13], [12] have worked on k-coverage problem and developed several algorithms, and heuristics to tackle the issue, none of the researchers have addressed the coverage imbalance problem that we introduce in this paper. We propose several solution ideas and a novel performance metric called "balancing index" to solve the problem. The major contributions of this paper can be summarized as follows:

- We introduce a novel *balanced k-coverage problem* for visual sensor networks (VSNs).

- We study the existing exact Integer Linear Programming (ILP) formulation for single coverage and extend the idea for k-coverage. Then we show that this natural extension provides imbalance coverage of the targets in a sense that some targets are covered k times while some targets are left totally uncovered or singly covered. The imbalance is more serious in *under-provisioned* networks.

- We propose a novel Integer Quadratic Programming (IQP) formulation that improves fairness by balancing coverage while trying to achieve k-coverage. We further improve the fairness by providing another novel Integer Non-Linear Programming (INLP) formulation.

- The proposed ILP, IQP, and INLP formulations are NP-complete and computationally expensive. Therefore, we formulate a novel computationally faster Centralized Greedy k-Coverage Algorithm (CGkCA) to approximate our formulations. Finally, we measure the relative performance of our formulations and the CGkCA algorithm in terms of coverage balancing.

3. Background

In this section we formally introduce the Visual Sensor Network along with relevant sensors parameters. We also elaborate how we determine if and to which pan of a sensor a target lies. We look into Fairness Index and its shortcomings and also introduce Balancing Index. We also provide a formal description of the Balanced k-Coverage Problem.

3.1. Visual Sensor Network Description and Parameters

The sensing region of a camera can be characterized by its Field of View (FoV) which is defined as the extent of the observable/sensing region that can be captured at any given direction. Some cameras come with fixed-FoV and for some, FoVs are adjustable. The smart cameras used in current VSNs are known as *Pan-Tilt-Zoom* (PTZ) cameras where FoV can be self-adjusted in

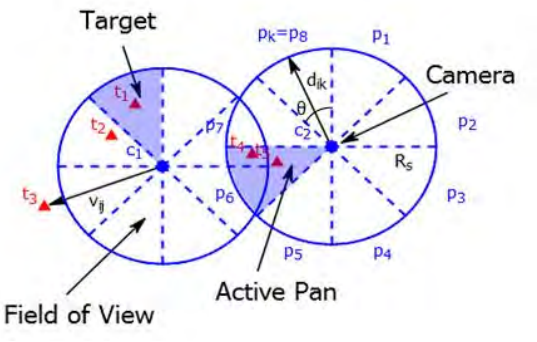


Figure 4: Camera coverage parameters

three dimensions: (i) horizontal movement in pan, (ii)vertical movement or tilt, and (iii) change in depth-of-field by changing zoom.

In this paper, we limit ourselves to pan-only cameras, i.e., we assume that a camera can move only in horizontal direction and its FoV is only described by its pan. The pan of a camera is formally defined using the following two parameters:

(a) R_s: Maximum coverage *range* of the camera beyond which a target can not be detected with acceptable accuracy in a binary detection test.

(b) θ: The maximum sensing/coverage *angle* of a camera on a certain direction. This angle is also known as Angle of View (AoV).

Thus, when a camera is oriented towards a particular direction, it can cover a circular sector (called a pan) defined by R_s and θ. We assume that every camera possesses a specific number of non-overlapping pans, of which, only one can be selected in a particular deployment. For example: a camera with FoV defined by $\theta = \frac{\pi}{4}$ can pick any one of eight disjoint orientations. Fig. 4 depicts these parameters of camera coverage. Here, two cameras c_1 and c_2 have eight pans each and can be oriented towards any of these eight pans. We assume that cameras are homogeneous in terms of parameters. Position of a target and a sensor are expressed through Cartesian coordinates (x, y) in a two-dimensional plane. $\overrightarrow{d_{ij}}$ is a unit vector which cuts each pan (i.e., the sensing sector) into half

representing the orientation of camera c_i towards pan p_j. $\overrightarrow{v_{it}}$ is a vector in the direction from camera c_i to target g_t.

3.1.1. Target in Sector (TIS) Test

With TIS test [3], one can verify whether a target g_t is coverable by a given sensor s_i. To conduct this test, at first we calculate the angle ϕ_{it} between camera orientation $\overrightarrow{d_{ij}}$ of pan p_j and the target vector $\overrightarrow{v_{it}}$.

$$\phi_{it} = cos^{-1}\left(\frac{\overrightarrow{d_{ij}} \cdot \overrightarrow{v_{it}}}{|\overrightarrow{d_{ij}}||\overrightarrow{v_{it}}|}\right) \tag{2}$$

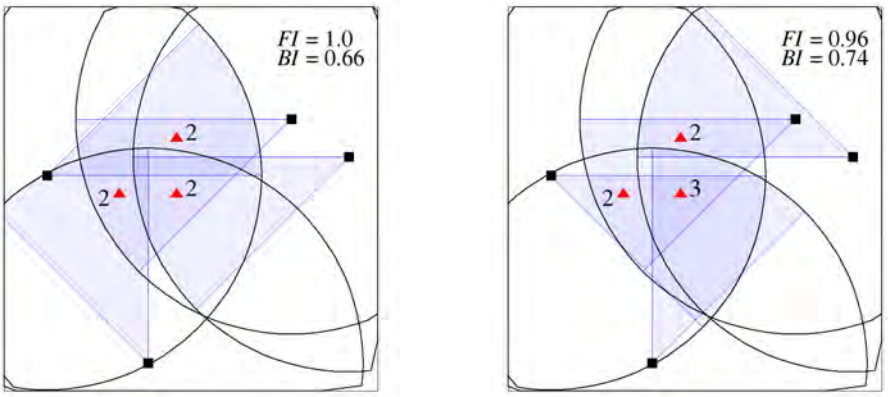
A target is coverable by a camera's FoV if the span of its FoV contains the target and the target is located within the sensing range of the camera. Geometrically, $\overrightarrow{d_{ij}}$ divides the pan p_j into two equal halves and if a target is located in either of them, it is coverable by that camera on the pan p_j. Thus, the angle ϕ_{it} needs to be less than half of the AoV, i.e., $\phi_{it} \leq \frac{\theta}{2}$. The other condition requires that the target has to be inside the maximum sensing range of the camera, i.e.,: $|v_{it}| \leq R_s$.

Conducting TIS tests over every pan p_j of camera s_i and every target g_t, we can build a *binary coverage matrix* $A^N_{M\times Q}$ of the network comprising of M targets and N cameras with Q pans where an entry in the matrix can be calculated as:

$$a^i_{tj} = \begin{cases} 1 & \text{if target } g_t \text{ is covered by camera } s_i \text{ at pan } p_j \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

3.2. Fairness Index and Balancing Index in k-coverage

Although the objective of the k-coverage problem is to cover each target at least k times, it may not be achievable if enough sensors are not available (i.e., in under-provisioned networks). Therefore, for under-provisioned networks a new metric needs to be defined to determine the superiority of one solution to another. The *Fairness index* proposed by Jain et al. [4] has been traditionally



(a) Isomorphic coverage (b) A better coverage

Figure 5: Demonstration of how BI works better than FI

used to measure *fairness* in resource allocation systems. The visual sensor network can be thought of a similar resource allocation system where the sensors are the resources and the targets are the components of the system. Suppose in a particular configuration and orientation of sensors, the m targets are covered $\psi_1, \psi_2, \ldots, \psi_m$ times respectively. Then the merit of this *solution vector* $(\psi_1, \psi_2, \ldots, \psi_m)$ can be measured using the following equation:

$$\mathcal{FI} = \frac{(\sum_{t=1}^{m} \psi_t)^2}{m \sum_{t=1}^{m} \psi_t^2} \tag{4}$$

Thus, the fairness index of two solution vectors $(3, 3, 1, 1)$ and $(2, 2, 2, 2)$ each of which tries to cover four targets in a 3-coverage problem is 0.8 and 1.0 respectively which shows the solution $(2, 2, 2, 2)$ is superior than $(3, 3, 1, 1)$.

Even though fairness index is a good performance metric, solely focusing on maximizing it can result in reducing the total coverage. Fairness index usually identifies fairer solutions in the allocation of resources. Thus, in a 3-coverage problem with 3 targets, (2, 2, 2) coverage is fairer than (2, 3, 2) although the second one is preferred because 3-coverage has not been attained at all in the solution (2, 2, 2) for any of the targets.

Therefore, we modify the concept of fairness index to fit in to our purpose and introduce a more suitable metric called *Balancing Index*, $\mathcal{BI}$, which combines both fairness of coverage and maximization of the total coverage. It is defined as the product of fairness index and the ratio of achieved total coverage (i.e., $\sum_{t=1}^{m} \psi_t$) over total attainable coverage (i.e., km). Mathematically, the balancing index is:

$$\mathcal{BI} = \mathcal{FI} \times \frac{\sum_{t=1}^{m} \psi_t}{km} = \frac{(\sum_{t=1}^{m} \psi_t)^2}{m \times \sum_{t=1}^{m} {\psi_t}^2} \times \frac{\sum_{t=1}^{m} \psi_t}{km} \tag{5}$$

where ψ_t is the number of sensors covering target g_t and m is the total number of targets.

In Fig. 5, there are 4 sensors (black squares) and 3 targets (red triangles). Fig. 5a and 5b shows two possible arrangements. In Fig. 5a, the coverage of targets are (2, 2, 2) and in Fig. 5b, the coverage of targets are (2, 3, 2). The balancing index value for (2, 2, 2) is:

$$\mathcal{BI} = \frac{(2+2+2)^2}{3 \times (2^2+2^2+2^2)} \times \frac{(2+2+2)}{3 \times 3} = 0.6666$$

The balancing index value for (2, 3, 2) is:

$$\mathcal{BI} = \frac{(2+3+2)^2}{3 \times (2^2+3^2+2^2)} \times \frac{(2+3+2)}{3 \times 3} = 0.7472$$

Fig. 5a has maximum fairness as all the targets have equal coverage, however the coverage of Fig. 5b is more preferable. The balancing index reflects that (2, 3, 2) coverage is better than (2, 2, 2) coverage in a 3-coverage problem. Thus, it can be used as the performance metric. The higher the value of the balancing index, the better is the coverage.

3.3. Problem Formulation

The balanced k-coverage problem can formally be described as follows:

Given: A set of targets, $\mathcal{T} = \{g_1, g_2, \ldots, g_m\}$ to be covered; a set of homogeneous directional sensors, $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$, each of which can be oriented in one direction of q possible non-overlapping pans; the pan set, $\mathcal{P} = \{p_1, p_2, \ldots, p_q\}$. A collection of subsets, $\mathcal{F} = \{\Phi_{\langle i,j \rangle} | s_i \in \mathcal{S}, p_j \in \mathcal{P}\}$ can be

generated based on a TIS test, where $\Phi_{\langle i,j\rangle}$ is a subset of $\mathcal{T}$ and denotes the set of targets covered by selecting sensor s_i and oriented in the direction of pan p_j. All the targets and the sensors are in the same two dimensional plane.

Problem: Find a sub-collection $\mathcal{Z}$ of $\mathcal{F}$, with the constraint that at most one $\Phi_{\langle i,j\rangle}$ can be chosen for the same sensor s_i, and the Balancing Index, $\mathcal{BI}$ (defined in Equation 5), gets maximized.

4. *k*-Coverage and its ILP Formulation

In [3], the authors, Ai and Abouzeid, elaborate an ILP formulation to solve the maximization of *single* coverage using minimum number of sensors (MCMS). The proposed ILP formulation can be easily extended for the k-coverage problem. In this section, at first we describe the necessary modifications and then we point out its shortcomings in providing a balanced solution for under-provisioned networks. This extension of Ai and Abouzeid's ILP formulation [3] has been compared later on with the proposed heuristics in the experimental results section.

4.1. Parameters

The parameters used for the formulation can be summarized as follows. n: the number of sensors; m: the number of targets; q: the number of orientation available for each directional sensor. The variables in the formulation are as follows. ψ_t: an integer variable that has a value equal to the number of times a target g_t is covered by directional sensors, limited up to a maximum value k; $\chi_{\langle i,j\rangle}$: a binary variable that has value one if the directional sensor s_i uses the orientation p_j, and zero otherwise; ξ_t: an integer variable that counts the number of the directional sensors covering target g_t. $\Phi_{\langle i,j\rangle}$ is the set of targets that are covered by the sensor s_i in its pan p_j. Using TIS, for each sensor s_i, incidence matrix $A^i_{(m\times q)}$ can be generated, where each of its elements would be:

$$a^i_{tj} = \begin{cases} 1 & t \in \Phi_{\langle i,j\rangle} \\ 0 & otherwise \end{cases} \tag{6}$$

Therefore, ξ_t can be expressed as:

$$\xi_t = \sum_{i=1}^{n}\sum_{j=1}^{p} a_{tj}^{i}\chi_{\langle i,j\rangle}$$

Now, the ILP formulation for k-coverage problem becomes:

$$\textbf{maximize} \quad \sum_{t=1}^{m}\psi_t - \rho\sum_{i=1}^{n}\sum_{j=1}^{q}\chi_{\langle i,j\rangle} \tag{7}$$

subject to:

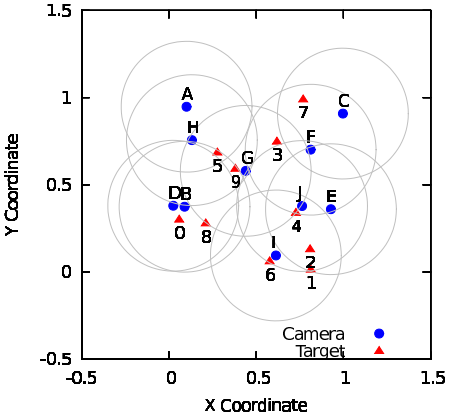
$$\frac{\xi_t}{n} \leq \psi_t \leq \xi_t \quad \forall t = 1\ldots m \tag{8}$$

$$\psi_t \leq k \tag{9}$$

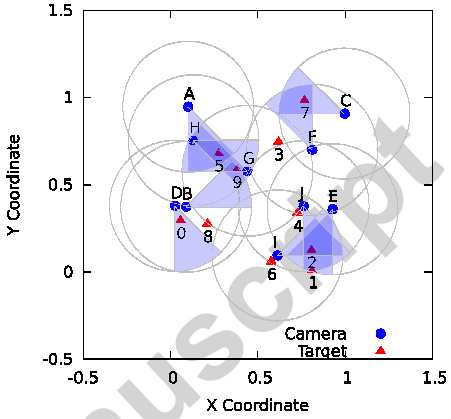
$$\sum_{j=1}^{q}\chi_{\langle i,j\rangle} \leq 1 \; \forall i = 1\ldots n \tag{10}$$

$$\chi_{\langle i,j\rangle} = 0 \quad or \quad 1 \quad \forall i = 1\ldots n, \; \forall j = 1\ldots q \tag{11}$$

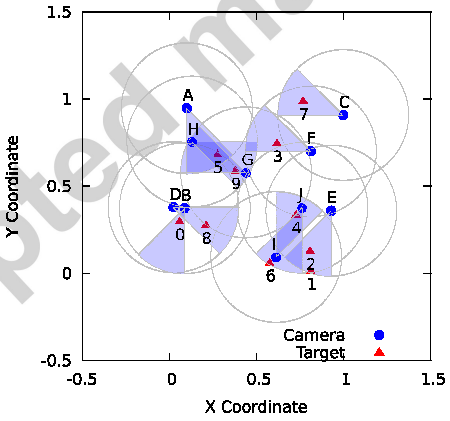
The objective function defined by Equation 7 maximizes the coverage count of each target and imposes a penalty by multiplying the number of sensors to be activated by a small positive penalty coefficient ρ (≤ 1). There are $(m + np)$ variables and $(2m + n + np)$ constraints for the ILP. Equation 8 represents a set of inequalities to indicate whether any target g_t is covered or not: if none of the sensors cover target g_t, i.e., $\xi_t = 0$, then $\psi_t = 0$ to conform the right inequality; if target g_t is covered by at least one directional sensor, i.e., $\xi_t > 0$, since ξ_t is bounded by n, ξ_t/n is a real number less than one, then $\psi_t \geq 1$ to follow the left inequality. Constraints in Equation 9 make sure that the coverage count of any target is bounded by k, i.e., even if a target is covered by more than k times still the coverage count will be considered as k, no additional benefit for covering a target more than k times. Equation 10 guarantees that one directional sensor has at most one orientation depending on whether it is activated or not.



(a) Initial Configuration 10 Targets and 10 Cameras



(b) Pan selection of each camera in ILP



(c) Pan selection in balanced-optimal configuration

Figure 6: Illustrating imbalance coverage of ILP formulation

4.2. Problem with generic ILP

The ILP formulation mentioned above for k-coverage problem does not focus on coverage balancing. To understand the problem, consider the scenario shown in Fig. 6a. Here we have shown a 3-coverage problem with 10 targets (red triangles) and 10 directional sensors (blue circles). FoV is defined by $\theta = \frac{\pi}{4}$. Table 1 summarizes the coverage counts under different conditions. Not all targets are 3-coverable:–it is possible to cover the targets $\{0, 3, 6, 7\}$ at most twice. Rest of the targets are at least 3-coverable. Clearly the network is under-provisioned. After running the ILP, formulated above, we found a solution which is shown in Fig. 6b. The third column of Table 1 captures the coverage achieved in ILP. Targets $\{2, 5, 9\}$ are covered thrice, targets $\{1, 7\}$ are covered twice, target $\{0\}$ is singly covered and noticeably targets $\{3, 4, 6, 8\}$ are left totally uncovered.

In summary, 40% targets are not covered by any of the sensors in the solution provided by the ILP. Fig. 6c shows another possible solution of the same problem which we call *balanced optimal* coverage (describe in Section 5.4). The fourth column of Table 1 shows the coverage achieved in this new solution. Only target $\{5\}$ and targets $\{2, 9\}$ are 3-covered and 2-covered respectively however the rest of all targets are singly covered. Unlike ILP, none of the targets are left uncovered in this new solution.

5. Balanced k-Coverage

In order to improve the balancing of coverage, we need to modify the ILP formulation, specially its objective function. In particular, the objective function concurrently needs to keep track of balancing of coverage while pursuing k-coverage. In this section we introduce two new formulations which will give much balanced solutions.

5.1. k-Coverage as Vectors

To keep the problem tractable, we may consider the solutions of k-coverage problem as vectors in an m-dimensional vector space. The coverage counts of

Table 1: Detail analysis of example scenario in Fig. 6

Target ID	Maximum possible coverage	Coverage achieved in ILP	Coverage achievable in Balanced-optimal
0	2	1	1
1	3	2	1
2	3	3	1
3	2	0	1
4	5	0	2
5	4	3	3
6	2	0	1
7	2	2	1
8	3	0	1
9	3	3	2

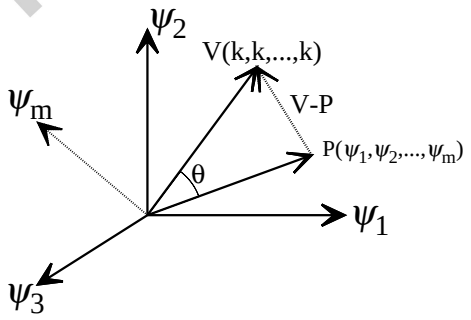


Figure 7: Coverage displayed in m-dimensional space

targets can be considered as individual dimensions in the m-dimensional space. Fig. 7 shows this typical scenario. Each axis represents the coverage count of each target and there is a total of m possible targets. The vector $\mathbf{V} \equiv (k, k, \ldots, k)$ represents the desired solution vector. Let us consider vector $\mathbf{P}$ which represents the achieved coverage by an arbitrary algorithm. Hence vector $\mathbf{P}$ can be represented as $\mathbf{P} \equiv (\psi_1, \psi_2, \ldots, \psi_k)$. Although it is highly desirable to align the vector $\mathbf{P}$ with the vector $\mathbf{V}$, however practically it may not be achieved by an algorithm. In such cases, the goal should be to *minimize* the *distance* between these two vectors, $\overrightarrow{\mathbf{PV}}$ [$\overrightarrow{\mathbf{PV}} = \mathbf{V} - \mathbf{P}$] or to *minimize* the *angle* between them (i.e., the θ). We formally describe both of these intuitive approaches below.

5.2. Minimizing the vector distance

The vector distance between the actual coverage vector $\mathbf{P}(\psi_1, \psi_2, \ldots, \psi_k)$ and the expected coverage vector $\mathbf{V}(k, k, \ldots, k)$ can be calculated as follows:

$$d(\mathbf{V}, \mathbf{P}) = ||\mathbf{V} - \mathbf{P}|| = \sqrt{\sum_{t=1}^{m} (k - \psi_t)^2}$$

where k is the number of times the targets are to be covered and ψ_t is the achieved coverage of target g_t. The minimization problem remains the same even if we ignore the square root. Thus, the ILP formulation described in Section 4 can be easily modified to achieve the goal. We can simply modify the objective function (Equation 7) and incorporate the square of the vector distance leaving all constraints (Equations 8 - 11) unchanged. The new objective function will be as follows:

$$\textbf{minimize} \quad \sum_{t=1}^{m} (k - \psi_t)^2 + \rho \sum_{i=1}^{n} \sum_{j=1}^{q} \chi_{\langle i,j \rangle}$$

The modified objective function is no more linear but quadratic in nature. Therefore, the new formulation is an *integer quadratic programming* problem or IQP in short.

5.3. Minimizing the angle

Another approach is to minimize the angle between the two vectors **V** and **P** using the following equation:

$$\theta = cos^{-1}\left(\frac{V.P}{||V||||P||}\right) = cos^{-1}\left(\frac{\sum_{t=1}^{m}\psi_t}{\sqrt{m\sum_{t=1}^{m}\psi_t^2}}\right)$$

$$\frac{\sum_{t=1}^{m}\psi_t}{\sqrt{m\sum_{t=1}^{m}\psi_t^2}} \Rightarrow \sqrt{\frac{(\sum_{t=1}^{m}\psi_t)^2}{m\sum_{t=1}^{m}\psi_t^2}} \Rightarrow \sqrt{\mathcal{FI}}$$

Therefore,

$$\theta = cos^{-1}\left(\frac{V.P}{||V||||P||}\right) = cos^{-1}\left(\frac{\sum_{t=1}^{m}\psi_t}{\sqrt{m\sum_{t=1}^{m}\psi_t^2}}\right) = \sqrt{\mathcal{FI}}$$

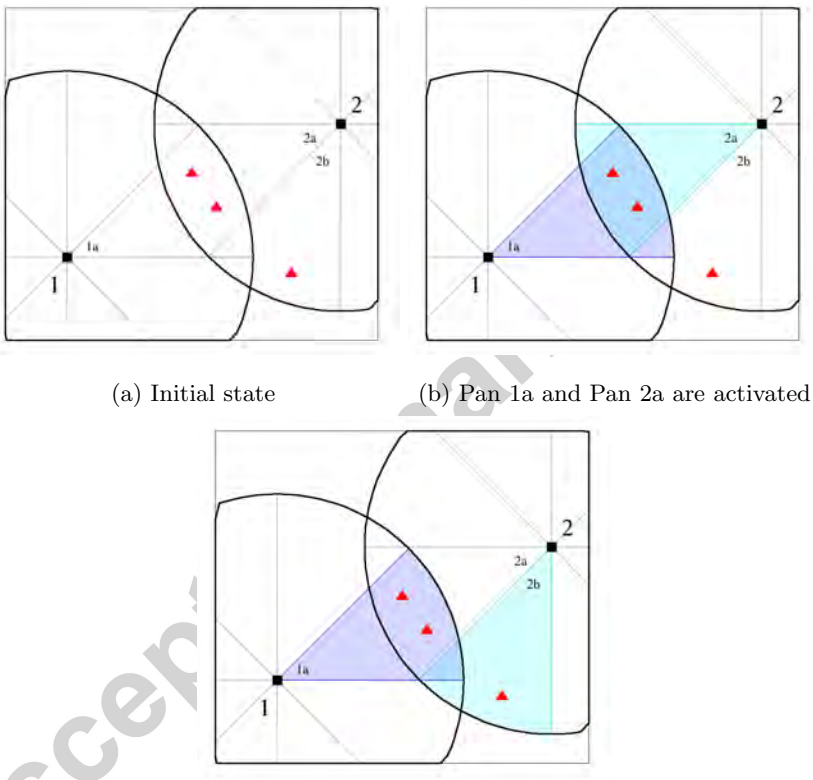
Therefore, it can be seen that minimizing angle depends on the fairness index. However, we have already seen the limitations of fairness index in Section 3.2. Minimizing the angle between vectors can not differentiate isomorphic solutions like $(1, 1, \ldots, 1)$, $(2, 2, \ldots, 2)$, $(3, 3, \ldots, 3)$ ... etc. because all of these vectors make an angle zero with the ideal solution vector $(k, k, \ldots, k)$. We do not explore this issue further in this paper.

5.4. Maximizing the balancing index

Finally, the objective function of the ILP can be modified to incorporate **balancing index** as defined in Section 3.2 and achieve true coverage balancing. The necessary modification is as follows:

$$\textbf{maximize} \quad \frac{1}{km^2} \times \frac{(\sum_{t=1}^{m}\psi_t)^3}{\sum_{t=1}^{m}\psi_t^2} - \rho\sum_{i=1}^{n}\sum_{j=1}^{p}\chi_{\langle i,j\rangle}$$

subjected to the constraints described in Equations 8 - 11. Note that, the new objective function is non-linear in nature. Therefore, the modified formulation falls within the domain of *integer non-linear programming* problems or INLP in short.



(a) Initial state (b) Pan 1a and Pan 2a are activated

(c) Pan 1a and Pan 2b are activated

Figure 8: Demonstration of Benefit Function

5.5. Comparative performance between IQP and INLP

We want to see how IQP and INLP behaves for a small scenario. In Fig. 8, there are 2 sensors (black rectangles) and 3 targets (red triangles). Fig 8b and Fig. 8c shows two possible orientations. Both figures have the same orientation for Camera 1 i.e., pan 1a. Camera 2 is oriented to pan 2a in Fig. 8b and pan 2b in Fig. 8c respectively. The value of $\mathcal{BI}$ for Fig. 8b is:

$$\mathcal{BI} = \frac{(2+2+0)^2}{3 \times (2^2+2^2+0^2)} \times \frac{(2+2+0)}{3 \times 3} = 0.296$$

The $\mathcal{BI}$ value of Fig. 8c is:

$$\mathcal{BI} = \frac{(1+1+1)^2}{3 \times (1^2+1^2+1^2)} \times \frac{(1+1+1)}{3 \times 3} = 0.333$$

IQP will have a cost of 11 for Fig. 8b and 12 for Fig.8c. Therefore, IQP will choose the orientation of Fig. 8b. Orientation in Fig. 8c has a greater $\mathcal{BI}$ value than that of Fig. 8b, and will be chosen by INLP.

6. Balanced k-Coverage Heuristics

The ILP, IQP, and INLP formulations of the problem mentioned in the previous section can be used to find the optimal solutions, however, they are not scalable in large problem instances. Therefore, we present Centralized Greedy k-Coverage Algorithm (CGkCA), a polynomial time greedy heuristic that would converge faster and by suitably choosing appropriate set of sensors would also balance the coverage. CGkCA uses a *benefit function*, which calculates the incentive of selecting a particular ⟨sensor, pan⟩ pair at each step.

6.1. Central Greedy k-Coverage Algorithm

The basic idea of CGkCA is to greedily choose and activate the ⟨sensor, pan⟩ pair of the inactive sensors which provides the maximum benefit. In each iteration, the incentives of all inactive ⟨sensor, pan⟩ pairs are calculated using the benefit function. The pair with maximum incentive is selected and the sensor is activated towards the corresponding pan. Ties are broken arbitrarily

Algorithm 1 Centralized Greedy k-Coverage Algorithm (CGkCA)

Input: $\Phi_{\langle i,j \rangle}$ {set of targets covered by sensor s_i in pan p_j}

Output: $\mathcal{Z}$ {a collection of $\langle$active sensor, orientation$\rangle$ pairs}

1: $\mathcal{Z} \leftarrow \emptyset$
2: $\mathcal{Y} \leftarrow$ {set of inactive nodes}
3: $\mathcal{T} \leftarrow \{g_1, g_2, \ldots g_m\}$ {set of all targets}
4: $\mathcal{C} \leftarrow \{c_1, c_2, \ldots c_m\}$ {c_i is the coverage count of g_t}
5: **repeat**
6: $\quad maxincentive \leftarrow 0$
7: $\quad \alpha \leftarrow 0$ for linear benefit; $\alpha \leftarrow 1$ for quadratic benefit
8: $\quad$ **for** $\forall i \in \mathcal{Y}$ **do**
9: $\quad\quad$ **for** $\forall j \in \mathcal{P}$ **do**
10: $\quad\quad\quad incentive \leftarrow$ BENEFIT$(\Phi_{\langle i,j \rangle}, \mathcal{Z}, \alpha)$
11: $\quad\quad\quad$ **if** $incentive > maxincentive$ **then**
12: $\quad\quad\quad\quad maxincentive \leftarrow incentive$
13: $\quad\quad\quad\quad \langle i^{max}, j^{max} \rangle = \langle i, j \rangle$
14: $\quad\quad\quad$ **end if**
15: $\quad\quad$ **end for**
16: $\quad$ **end for**
17: $\quad \mathcal{Z} \leftarrow \mathcal{Z} \cup \langle i^{max}, j^{max} \rangle$
18: $\quad \mathcal{Y} \leftarrow \mathcal{Y} \backslash \{i^{max}\}$
19: **until** $maxincentive = 0$
20: **return** $\mathcal{Z}$

by choosing among the pairs providing maximum incentive. The algorithm terminates when all the sensors are activated or when all the targets are at least k-covered. As CGkCA chooses these ⟨sensor, pan⟩ pairs greedily and does not backtrack, the algorithm is susceptible to produce suboptimal solutions. The pseudo code of CGkCA is given in Algorithm 1.

6.2. Benefit function

In order to choose the immediate best possible sensor-pan pair from all inactive sensor-pan pairs, CGkCA makes a call to a benefit function in step 10 of Algorithm 1. The benefit function, which is defined in Algorithm 2, calculates two different kinds of benefit: (i) linear benefit, and (iii) quadratic benefit.

The three parameters of the benefit function is as follows:

- $\Phi_{\langle i,j \rangle}$: set of targets covered by sensor-pan pairs $\langle i, j \rangle$.
- $\mathcal{Z}$: set of sensor-pan pairs activated so far by the greedy algorithm before this step.
- α: a boolean parameter to indicate whether the benefit function should calculate either linear or quadratic benefit. $\alpha = 0$ means linear and quadratic otherwise.

The linear benefit is calculated as follows. For each target g_t, let us define a variable c_t which assumes a value equal to the coverage count of that target by the set of sensor-pan pairs in set $\mathcal{Z}$, i.e.:

$$c_t = \text{number of sensing regions in } \mathcal{Z} \text{ that cover target } g_t$$

Then the total linear benefit of activating sensor-pan pair $\langle i, j \rangle$ at this stage of CGkCA is:

$$\sum_{t \in \Phi_{\langle i,j \rangle} \wedge c_t < k} min\{1, k - c_t\}$$

This approach is a special case of the *greedy algorithm* proposed in [20] for directional sensors with overlapping pans. We name their approach as *Greedy*

Algorithm 2 Benefit Function for k-coverage

Input: $\Phi_{\langle i,j \rangle}$ {set of targets covered by sensor s_i in pan p_j}, $\mathcal{Z}$ {set of assigned $\langle$active sensor, orientation$\rangle$ pairs}, α {$\alpha = 0$ for linear benefit and $\alpha = 1$ for quadratic benefit}

Output: *incentive* {an integer containing the total incentive for $\langle i, j \rangle$ for given $\mathcal{Z}$}

1: **function** BENEFIT($\Phi_{\langle i,j \rangle}, \mathcal{Z}, \alpha$)
2: $incentive \leftarrow 0$
3: **for** $\forall t \in \Phi_{\langle i,j \rangle}$ **do**
4: Calculate the coverage c_t of target t using $\mathcal{Z}$
5: **if** $c_t < k$ **then**
6: **if** $\alpha = 0$ **then**
7: $increment \leftarrow 1$
8: **else if** $\alpha = 1$ **then**
9: $increment \leftarrow (k - c_t)^2 - (k - c_t - 1)^2$
10: **end if**
11: $incentive \leftarrow incentive + increment$
12: **end if**
13: **end for**
14: **end function**
15: **return** *incentive*

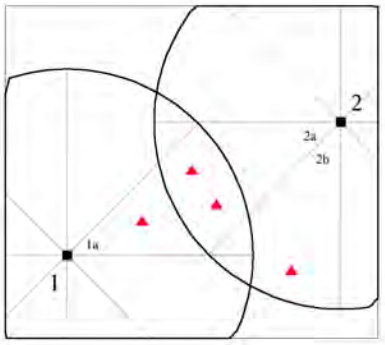
Table 2: Incentive table for 3-coverage problem ($k = 3$)

Coverage count of a target at this stage	Incentive in linear benefit function	Incentive in quadratic benefit function
c_t	$min\{1, k - c_t\}$	$(k - c_t)^2 - (k - c_t - 1)^2$
0	1	5
1	1	3
2	1	1

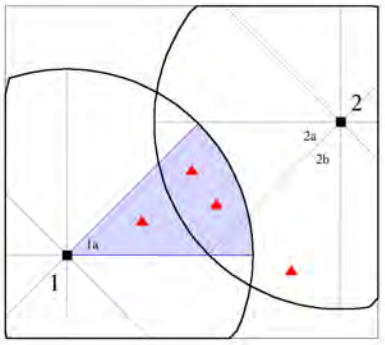
Linear and compare its performance with the proposed heuristics in the experimental result section later on. Note that, the linear benefit function ignores the coverage count of a target at this stage of the greedy algorithm and assigns same incentive for covering a target which is less covered or has not been covered at all with respect to for covering a target which is already covered by many sensors irrespective of its coverage count. The quadratic benefit function eliminates this drawback and provides more incentive for covering a less covered target as opposed to highly covered targets. The incentive of covering a target is quadratic in nature and is defined by $k^2 - (k - c)^2$ for a target that is covered c times. Thus the quadratic benefit of activating sensor-pan pair $\langle i, j \rangle$ at this stage of CGkCA becomes:

$$\sum_{t \in \Phi_{\langle i,j \rangle} \wedge c_t < k} (k - c_t)^2 - (k - c_t - 1)^2$$

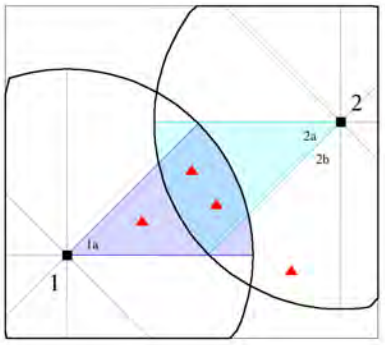
Table 2 shows the incentive values for both linear and quadratic benefit functions that will be rewarded while solving a 3-coverage problem. Algorithm 2 provides the complete pseudo-code of the benefit function. For rest of the paper, when the greedy algorithm runs with the linear benefit function, the approach is dubbed as *Greedy Linear*. Similarly, when it runs with quadratic benefit function, we call it *Greedy Quadratic*.



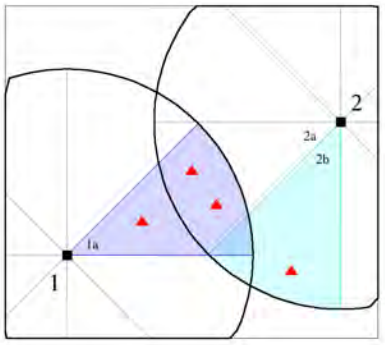
(a) Initial state



(b) Greedy Linear and Greedy Quadratic choose same camera-pan pair



(c) Greedy Linear activates pan 2a



(d) Greedy quadratic chooses pan 2b

Figure 9: Demonstration of Benefit Function

Table 3: Incentive value initially for Fig. 9a

Camera-Pan pair	Incentive in linear benefit function	Incentive in quadratic benefit function
1a	3	9
2a	2	6
2b	1	3

Table 4: Incentive value after Camera 1 is assigned, Fig 9b

Camera-Pan pair	Incentive in linear benefit function	Incentive in quadratic benefit function
2a	2	2
2b	1	3

6.2.1. Benefit function at work

In this section, we demonstrate how benefit function works. We will see how the difference of incentive schemes in benefit function produces different solutions. We will also see how Greedy Quadratic produces a more balanced solution than that of produced by Greedy Linear. In Fig. 9 shows 2 cameras (black squares), numbered 1 and 2, and 4 targets (red triangles) and it is a 2-coverage problem. As targets lie only on in 1 pan of Camera 1 and in 2 pans of Camera 2, we need to consider only these three pans. Initially, none of the targets are covered.

Table 3 shows the incentive values for both Greedy Linear and Greedy Quadratic. Both heuristics would choose camera-pan 1a to activate, as shown in Fig. 9b. Table 4 shows the incentive values after Camera 1 has been assigned to 1a. As none of the targets have been 2-covered, the incentive using linear benefit function does not change for the remaining 2 camera-pan pairs. However, as 3 of the targets are once covered, the incentive values, calculated

for Greedy Quadratic, for covering those sensor will be less. Greedy Linear will choose to activate Camera 2's 2a pan, as can be seen in Fig. 9c. Pan 2b has the greater incentive for Greedy Quadratic and it will be chosen, as in Fig. 9d.

The $\mathcal{BI}$ value of Fig. 9c is:

$$\mathcal{BI} = \frac{(1+2+2+0)^2}{4 \times (1^2+2^2+2^2+0^2)} \times \frac{(1+2+2+0)}{2 \times 4} = 0.434$$

The $\mathcal{BI}$ value of Fig. 9d is:

$$\mathcal{BI} = \frac{(1+1+1+1)^2}{4 \times (1^2+1^2+1^2+1^2)} \times \frac{(1+1+1+1)}{2 \times 4} = 0.5$$

Thus, it can be seen that Greedy Quadratic offers more balanced coverage than that of Greedy Linear.

6.3. Time complexity

CGkCA needs the incidence matrix as input. To generate the incidence matrix, we need to iterate over all the n sensors in each of their q pans and check for each target if they satisfy the TIS test. As the TIS test takes $O(1)$ time for a specific sensor in its specific pan and a fixed target, the whole generation would take $O(nmq)$ time.

The major contributor to time complexity of the algorithm is due to the calculation of maximum incentive. The loop from Step 5 to Step 19 is executed at most n times to check for each sensor. In each iteration, the benefit function will be called $O(nq)$ times. On each call, benefit function will check all the target within that pan and calculate the coverage counts of those targets and then the incentive of the ⟨sensor, pan⟩ pair. It will cost $O(mn)$. Therefore, the cost of each iteration is $O(n^2qm)$. Thus, the overall time complexity of the algorithm becomes $O(n^3qm)$.

7. Experimental Results

In order to verify and compare the effectiveness of proposed ILP, IQP, and INLP formulations and the greedy algorithm with linear and quadratic benefit

functions, we perform rigorous simulation experiments. We use *balancing index*, $\mathcal{BI}$, defined in Section 3.2 as the performance metrics of comparison. A higher value of $\mathcal{BI}$ indicates highly balanced coverage. At the end, we also comment on the sensor usage by the different formulations and greedy heuristics.

7.1. Simulation Environment

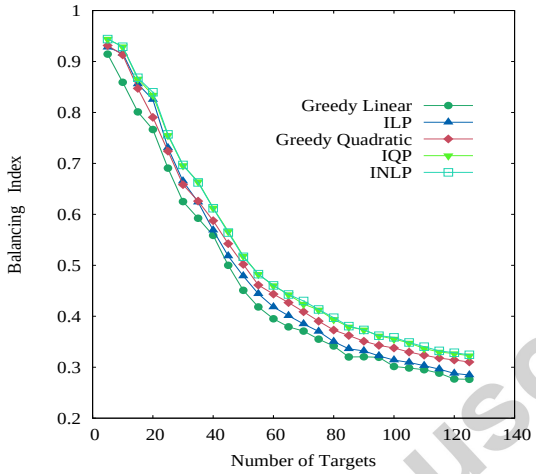
The deployment area is modelled as a 2D grid where targets are considered points in the grid and the sensors are modelled as directional sensors with FoV, $\theta = \frac{\pi}{4}$. We run two different types of experiments. In one kind, we keep the number of sensors fixed at 50 and vary the number of targets from 5 to 125 and in another kind, we keep the number of targets fixed at 50 and vary the number of available sensors from 20 to 115. For both scenarios the sensing range $R_s = 25$ units and the grid size is 125×125 sq. units. In both cases, the scenarios are generated in such a way that the smaller scenario is a subset of a larger scenario. This ensures a consistent evaluation of the impact of the enlarged population of sensors/targets by retaining all the "features" of the previous environment and simply making it better/worse.

7.2. Performance comparison of different approaches

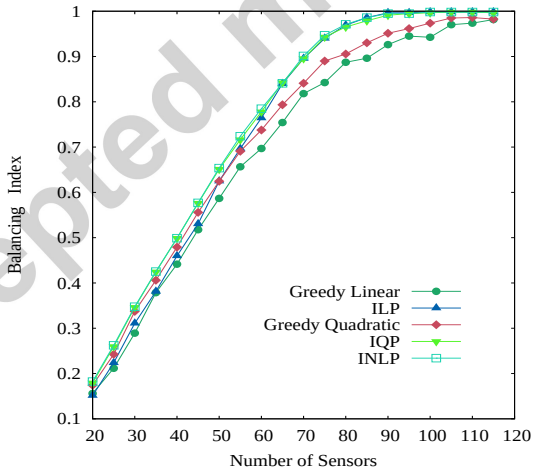
We capture various performance analysis of the proposed approaches such as- $\mathcal{BI}$, coverage analysis while changing the network's state from under-provisioned to over-provisioned type and vice-versa. We also approximate energy consumption by analyzing sensor usage. Firstly, for a given scenario we find out the incidence matrix. We modelled the IQP and ILP formulations in C++ and use CPLEX [21] optimizer for solving them. CPLEX optimizer can neither be used for non-linear objective function nor for cubic or higher degree objective functions. Therefore, we model the INLP formulations in AMPL [22] and solve the models using Couenne [23],[24] optimizer.

7.2.1. Using $\mathcal{BI}$ as performance metric

In the first type of environment, when we increase the number of targets gradually from 5 to 125 keeping number of sensors fixed at 50, the network's



(a) Performance comparison while varying the number of targets; the number of sensors is fixed at 50



(b) Performance comparison while varying the number of sensors, the number of targets is fixed at 50

Figure 10: Effect on Balancing Index

state changes from over-provisioned to under-provisioned as the number of targets slowly overwhelms the number of sensors. As a result, from Fig. 10a it is clearly evident that the curves move farther away from the ideal coverage ($\mathcal{BI} = 1$). In the whole downward progress, INLP and IQP clearly outperform all other methods.

In the second type of environment, increasing the number of available sensors and keeping the number of targets fixed at 50 shifts the state of the network from under-provisioned to over-provisioned. This behaviour is reflected in all curves as there is an upward movement toward the ideal coverage in Fig. 10b. Also here, the performance of INLP and IQP formulations exceeds the other approaches.

In both cases, the Greedy Quadratic shows greater coverage and balancing than the ILP formulation in under-provisioned condition, however when the network starts to contain larger number of sensors relative to the number of targets, ILP crosses the Greedy Quadratic curve. In a completely over-provisioned scenario, ILP almost merges with IQP and INLP formulations. Greedy Linear fails to keep up with all other formulations in all cases. One notable point is that, IQP curve is almost merged with INLP curves throughout the whole simulation however never exceeds the performance of INLP. We can conclude that the Greedy Quadratic approximates the optimal behavior very closely and with a very reasonable amount of computational time.

7.2.2. Comparison in Required Computational Power

We consider the second type of scenario where the number of targets were kept fixed at 50. We see that from Fig. 11 the computation time required to calculate the optimal solutions from IQP and INLP increases as the number of sensors increases. For a fairly dense scenario where total number of sensors is at least 100, INLP requires more than a hundred seconds and IQP requires more than 50 seconds. ILP requires less than 10 seconds at most. Therefore, it is clear that these solutions are not fast enough to use in large scale scenarios involving hundreds of sensors and targets. On the other hand both the Greedy Linear

and Greedy Quadratic converges within a second, thus making these heuristics more suitable for practical deployment.

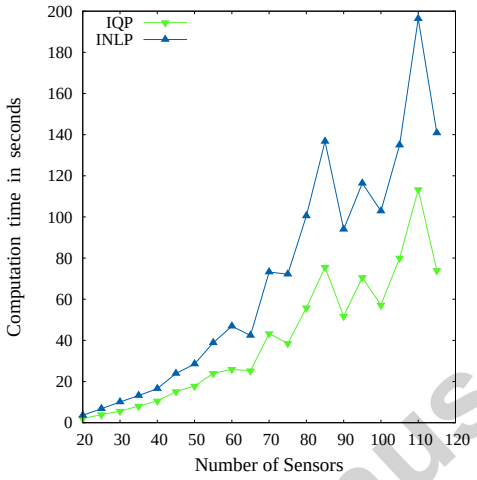
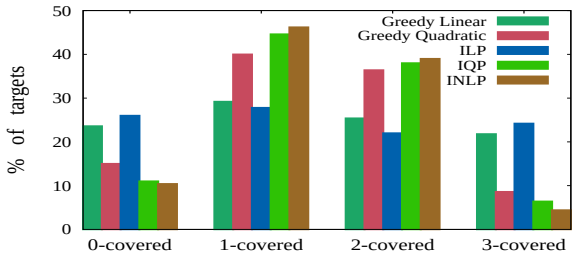


Figure 11: Comparison of Computational Time required by IQP and INLP

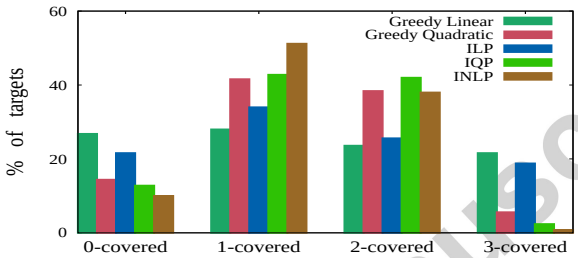
7.2.3. Coverage Analysis

Now we present a detailed coverage analysis of both under-provisioned and over-provisioned networks for some representative cases from our generated scenarios.

Under-provisioned Networks. In the scenario, where number of sensors were fixed, we consider the case in which the number of targets is 100 and the number of sensors is 50, as in Figure 12a. On the other scenario, as in Figure 12b, we consider the case where the number of targets is kept fixed at 50 and the number of sensors is 30. In both cases there are not enough sensors available to cover all the targets k-times ($k = 3$). We see that INLP as well as IQP tries to reduce the number of 0-covered targets with the expense of number of higher covered targets. The percentage of 0-covered targets is 10.4% and 11% for the first scenario, whereas it is 10% and 12.8% for INLP and IQP respectively in the second scenario. The Greedy Quadratic algorithm roughly approximates this behaviour by reducing the number of 0-covered targets in first scenario to 15%



(a) Number of sensors=50, number of targets=100

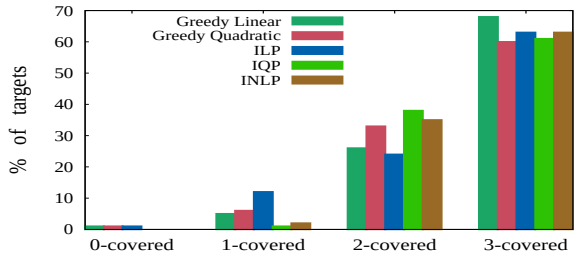


(b) Number of sensors=30, number of targets=50

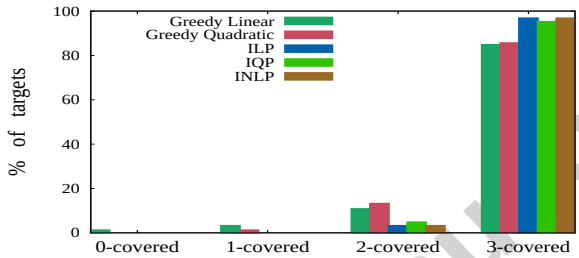
Figure 12: Coverages in Under-provisioned Network

and in the second scenario to 14.4%. The other two approaches (ILP and Greedy Linear) does not focus on balancing and as a result they increase the coverages of some targets, however keep larger number of targets totally uncovered.

Over-provisioned Networks. We consider two representative cases of over-provisioned network in both types of environment. In the first type, the number of sensors was kept fixed at 50, Figure 13a, and we consider the case where the number of targets is 20. In the later type, the number of targets was kept fixed at 50 and our representative case has 85 available sensors, as shown in Figure 13b. Clearly this is an over-provisioned network since there are enough available sensors to cover most of the targets at least k times. INLP and IQP formulations again gives more importance to targets covered lower number of times. As a consequence, in both cases there is no 0-covered targets using these two formulations. Due to abundance of available sensors, all the formulations tries to increase the number of 2-covered and 3-covered targets and INLP and IQP formulations



(a) Number of sensors=50, number of targets=20



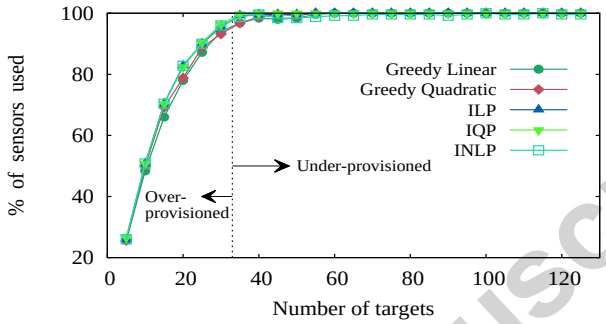
(b) Number of sensors=85, number of targets=50

Figure 13: Coverages in Over-provisioned Network

exceeds the other approaches in doing so. Among the greedy approximation algorithms, the quadratic approximation performs much better as it reduces the number of lower covered targets.

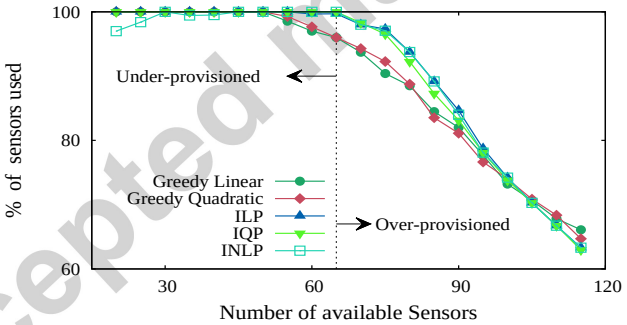
7.2.4. Sensor Usage Analysis

Fig. 14a shows the percentage of sensor usage for the scenarios of Fig. 10a. Percentage of sensors used gradually increases with the number of targets until all the cameras are activated. All the formulations perform almost similarly. Fig. 14b is the sensor usage diagram for scenarios in Fig. 10b. As the number of sensors gradually increases, lesser percentage of available sensors are activated. Although the number of sensors used by all formulations are similar when the number of available sensors is lower, the situation changes with the increase in available sensors. With 50 to 115 sensors, sensor usage of Greedy Linear and Greedy Quadratic is less than others. However in terms of coverage, they were always outperformed by IQP and INLP.



Number of sensors fixed at 50

(a) Sensor usage while varying number of targets



Number of targets fixed at 50

(b) Sensor usage while varying number of sensors

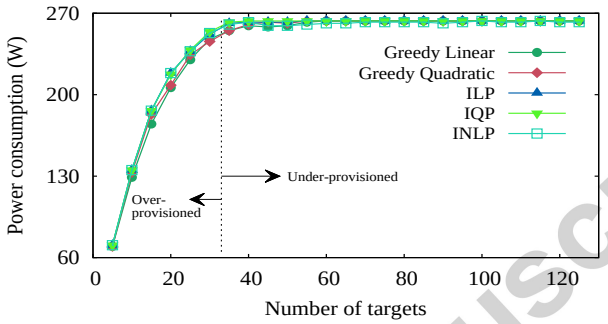
Figure 14: Analysis of Sensors usage

Interestingly, both Fig. 14a and Fig. 14b shows a clear transition from under-provisioned network to over-provisioned network and the sensor usage phenomena also changes accordingly. In Fig. 14a when the number of targets exceeds 50 and in Fig. 14b when the number of sensors falls below 55, the networks become under-provisioned. We can see that all of our formulations used up almost all the sensors in such under-provisioned networks. As a result all the curves become almost linear and parallel to the x-axis in these regions. Reducing sensor usage only happened in the over-provisioned networks (i.e., the other side of the plots). In this region, although all the formulations used almost the same number of sensors as shown in Fig. 14a, however INLP and IQP had much better coverage balancing over other formulations (see Fig. 10a). Among the variants of the greedy algorithm, Greedy Quadratic shows better coverage balancing over Greedy Linear, although they use the same number of sensors. From Fig. 14b and Fig. 10b, we can draw a similar conclusion.

7.2.5. Power Consumption

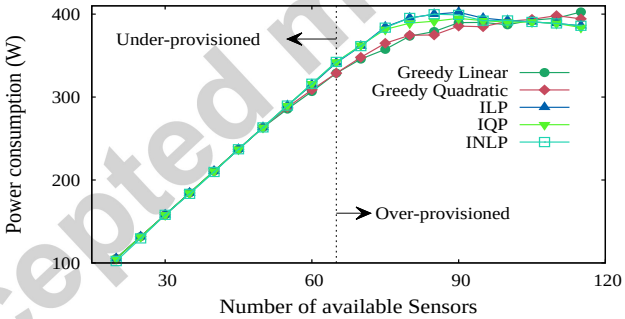
In order to calculate the power consumption of the proposed heuristics, we use the same model that has been used in [25]. In this model, a visual sensor can be in either one of the three possible states: *active*, *idle*, and *sleep*. In *active state*, the sensor monitors the targets and/or transmits the gathered information. In *idle state*, the sensor is powered up, however, it is neither monitoring nor transmitting, it simply runs the background OS processes. In *sleep state*, the sensor deactivates all of its hardware components. Among these three states, the active state is the *most* and the sleep state is the *least* power consuming state. Sensors can easily switch between active, idle and sleep states as needed. For all the heuristics, it has been assumed that an activated sensor remains active and continues to monitor the targets until its power drains off completely. The rest of the deactivated sensors remain in the sleep state and may become activated on-demand to replace any power-drained sensors. Therefore, the total power consumption, P_h, of a heuristic can be calculated as follows:

$$P_h = (number\ of\ active\ sensors) * P_a + (number\ of\ inactive\ sensors) * P_s$$



Number of sensors fixed at 50

(a) Power consumption while varying number of targets



Number of targets fixed at 50

(b) Power consumption while varying number of sensors

Figure 15: Power Consumption Analysis

where, P_a is the power consumption of a sensor in active state and P_s is the power consumption of the sensor in sleep state. For a Panoptes video sensor [26], the value of P_a and P_s is $5.268W$ and $0.058W$ respectively. We use these values for all our simulation results.

Fig. 15a and 15b are the power consumption graphs of Fig. 14a and 14b respectively. As the number of targets increases gradually from 5 to 125 keeping number of sensors fixed at 50, the system moves from over-provisioned condition to under-provisioned condition. In over-provisioned condition the number of sensors are more than enough to cover all the available targets, so a fraction of them is used and becomes active. The rest of the sensors remain inactive in sleep state. With addition of more targets, more and more sensors switches from sleep state to active state and naturally the power consumption gradually increases (see Fig. 15a). The curve trend is similar for all heuristics and ILP. The power graph becomes invariant in under-provisioned scenarios (scenarios with number of targets 30 or more) where there do not exist enough sensors to cover all the targets. All the heuristics perform similarly in under-provisioned systems where all 50 sensors are used up and the power consumption curves become constant at a certain level.

In Fig. 15b, we vary the number of sensors from 20 to 115 while keeping the number of targets fixed at 50 and the system switches from under-provisioned to over-provisioned system. At the beginning of curves the system is under-provisioned. From one scenario to another when gradually more sensors are added, the newly added sensors immediately gets activated as the system is under-provisioned. Thus, the power consumption increases almost linearly at the beginning. However, after adding a certain number of sensors (i.e., after adding 60 sensors) the network becomes over-provisioned where 3-coverage is almost already attained, and the newly added sensors are not needed any more leaving them under sleep state. Therefore, the power consumption becomes almost constant in over-provisioned networks; the little changes in the total power consumptions are due to the power consumed by the additional inactive sensors consuming only sleep state power.

8. Conclusions and Future Works

As low cost video-transmitting sensors are becoming more and more available, the opportunity to deploy them in fields has also been increased. In many cases, k-coverage is required over single coverage due to the required robustness and redundancy of the system. Our work addresses a novel problem of coverage imbalance in k-coverage of VSN. Coverage imbalance is a serious problem in under-provisioned networks, where the networks does not have enough sensors to ensure k-coverage of all the targets. We extended the traditional ILP designed for single coverage and applied it to solve the multi-coverage problem. However, the extension does not balance the coverage of the targets. We further designed quadratic (IQP) and non-linear (INLP) version of the ILP that are capable of addressing the coverage balancing. IQP minimizes the vector distance between the attained and expected coverage. INLP maximizes the Balancing Index ($\mathcal{BI}$) which is the product of Fairness Index ($\mathcal{FI}$) and average coverage. As ILP, IQP, and INLP are not scalable for large problem instances, we developed a greedy approach, CGkCA with two variants of incentive mechanism namely Greedy Linear and Greedy Quadratic. Even though both greedy approaches are outperformed by optimal algorithms, in under-provisioned networks Greedy Quadratic closely approximates the optimal solutions. We ran computer simulations to verify efficacy of the proposed formulations.

In future, we plan to carry out more simulations using different scenarios, such as different grid sizes and different sensing ranges. We also plan to run CGkCA under different incentive mechanisms and in real test beds. We used centralized heuristics, which might not always be usable in practical implementations. In the future, we plan to explore distributed approaches. By changing the value of penalty factor, ρ (> 1), of the objective functions, the formulations can be modified to reflect expected behavior of energy scarce networks. In certain applications, such as surveillance systems, there might not be specific target but rather the total region that needs to be k-covered. For such cases, we need to restructure our proposed solution for finding k-coverage of the area.

We have restricted our work to Pan only cameras, we need to explore possible balanced k-coverage formulations for Pan-Tilt-Zoom (PTZ) cameras.

References

[1] G. Fan, S. Jin, Coverage problem in wireless sensor network: A survey, Journal of Networks 5 (9) (2010) 1033–1040.

[2] C.-F. Huang, Y.-C. Tseng, The coverage problem in a wireless sensor network, Mobile Networks and Applications 10 (4) (2005) 519–528.

[3] J. Ai, A. A. Abouzeid, Coverage by directional sensors in randomly deployed wireless sensor networks, Journal of Combinatorial Optimization 11 (1) (2006) 21–41.

[4] R. Jain, D.-M. Chiu, W. Hawe, A quantitative measure of fairness and discrimination for resource allocation in shared computer systems.

[5] L.-H. Yen, C. W. Yu, Y.-M. Cheng, Expected k-coverage in wireless sensor networks, Ad Hoc Networks 4 (5) (2006) 636–650.

[6] M. Hefeeda, M. Bagheri, Randomized k-coverage algorithms for dense sensor networks, in: INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE, IEEE, 2007, pp. 2376–2380.

[7] H. M. Ammari, S. K. Das, A study of k-coverage and measures of connectivity in 3d wireless sensor networks, Computers, IEEE Transactions on 59 (2) (2010) 243–257.

[8] S. Kumar, T. H. Lai, J. Balogh, On k-coverage in a mostly sleeping sensor network, in: Proceedings of the 10th annual international conference on Mobile computing and networking, ACM, 2004, pp. 144–158.

[9] M. R. Garey, D. S. Johnson, Computers and intractability, Vol. 29, wh freeman, 2002.

[10] H. Brönnimann, M. T. Goodrich, Almost optimal set covers in finite vc-dimension, Discrete & Computational Geometry 14 (1) (1995) 463–479.

[11] Y. Bejerano, Simple and efficient k-coverage verification without location information, in: INFOCOM 2008. The 27th Conference on Computer Communications. IEEE, IEEE, 2008.

[12] C. Qiu, H. Shen, K. Chen, An energy-efficient and distributed cooperation mechanism for k-coverage hole detection and healing in wsns, in: Mobile Ad Hoc and Sensor Systems (MASS), 2015 IEEE 12th International Conference on, IEEE, 2015, pp. 73–81.

[13] K. M. Alam, J. Kamruzzaman, G. Karmakar, M. Murshed, Dynamic adjustment of sensing range for event coverage in wireless sensor networks, Journal of Network and Computer Applications 46 (2014) 139–153.

[14] D. Tian, N. D. Georganas, A coverage-preserving node scheduling scheme for large wireless sensor networks, in: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, ACM, 2002, pp. 32–41.

[15] B. Liu, D. Towsley, A study of the coverage of large-scale sensor networks, in: Mobile Ad-hoc and Sensor Systems, 2004 IEEE International Conference on, IEEE, 2004, pp. 475–483.

[16] C. B. Margi, V. Petkov, K. Obraczka, R. Manduchi, Characterizing energy consumption in a visual sensor network testbed, in: Testbeds and Research Infrastructures for the Development of Networks and Communities, 2006. TRIDENTCOM 2006. 2nd International Conference on, IEEE, 2006, pp. 8–pp.

[17] S. Lu, S. Yu, A fuzzy k-coverage approach for rfid network planning using plant growth simulation algorithm, Journal of Network and Computer Applications 39 (2014) 280–291.

[18] S. K. Gupta, P. Kuila, P. K. Jana, Genetic algorithm approach for k-coverage and m-connected node placement in target based wireless sensor networks, Computers & Electrical Engineering.

[19] V. P. Munishwar, N. B. Abu-Ghazaleh, Coverage algorithms for visual sensor networks, ACM Transactions on Sensor Networks (TOSN) 9 (4) (2013) 45.

[20] G. Fusco, H. Gupta, Selection and orientation of directional sensors for coverage maximization, in: Sensor, Mesh and Ad Hoc Communications and Networks, 2009. SECON'09. 6th Annual IEEE Communications Society Conference on, IEEE, 2009, pp. 1–9.

[21] I. ILOG, Cplex 12.4, http://www-01. ibm. com/software/integration/optimization/cplex-optimizer/.

[22] R. Fourer, D. Gay, B. Kernighan, AMPL: a modeling language for mathematical programming, Vol. 119, Boyd & Fraser, 1993.

[23] P. Belotti, J. Lee, L. Liberti, F. Margot, A. Wächter, Branching and bounds tighteningtechniques for non-convex minlp, Optimization Methods & Software 24 (4-5) (2009) 597–634.

[24] R. Lougee-Heimer, The common optimization interface for operations research: Promoting open-source software in the operations research community, IBM Journal of Research and Development 47 (1) (2003) 57–66.

[25] S. Farzana, K. A. Papry, A. Rahman, R. Rab, Maximally pair-wise disjoint set covers for directional sensors in visual sensor networks, in: 2016 Wireless Days (WD), IEEE, 2016, pp. 1–7.

[26] W.-c. Feng, E. Kaiser, W. C. Feng, M. L. Baillif, Panoptes: scalable low-power video sensor networking technologies, ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 1 (2) (2005) 151–167.

Sakib Md. Bin Malek received his BSc. degree in Computer Science and Engineering from the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh in 2015. His research interests include sensors networks, embedded systems, big data, and artificial neural networks. He is working as a Software Engineer in QI Analysis Inc, USA.

Md. Muntakim Sadik received his BSc. Engg degree in Computer Science and Engineering from the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh. After finishing his BSc he is working as a Research and Development Engineer in Works Applications PTE Ltd, Singapore. His research interests include sensor networks, neural networks, software development process and test automation.

Ashikur Rahman received his BSc and MSc degrees in Computer Science and Engineering from the Department of Computer Science and

Engineering, Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh in 1998 and 2001, respectively. He received his PhD in 2006 from the Department of Computing Science, University of Alberta, Canada. After finishing his PhD, he worked as a postdoctoral researcher at Simon Fraser University (2007), University of Calgary (2010-2011), McGill University (2011), Canada and Binghamton University (2012), USA. His research interests include ad-hoc and sensor networks, peer-to-peer computing, swarm intelligence, data center networks, cyber physical systems, back-end compiler optimization and neural networks. He is currently working as a Professor in the Department of Computer Science and Engineering, BUET.