2009

Network flow formulation of optimal perimeter sensory coverage problem

Davood Golmohammadi, PhD, *University of Massachusetts Boston*

European Journal of Operational Research 197 (2009) 77–83

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

Network flow formulation of optimal perimeter sensory coverage problem

Mohsen A. Jafari [a,*], Jiachen Liu [a], Davood Golmohammadi [b]

[a] Department of Industrial and Systems Engineering Rutgers, The State University of New Jersey, P.O. Box 909, Piscataway, NJ 08854, USA
[b] Department of Management Science and Information Systems, University of Massachusetts Boston, Boston, MA 02125, USA

ARTICLE INFO

Article history:
Received 10 August 2006
Accepted 19 June 2008
Available online 26 June 2008

Keywords:
Target tracking
Optimization
Network flow
Sensors

ABSTRACT

In this article, the perimeter detection optimization problem in field surveillance and target tracking are discussed. The detection range of sensors is assumed to be circular or elliptical. Sensors are also assumed to be associated with a cost factor reflecting their operational characteristics and power usage. We show that the problem of optimal sensor selection can be reduced to a network flow problem and can then be solved using any existing classical methodology. This significantly reduces the computational time of sensory selection problem which in many cases needs to be solved in almost real time basis, every time that the dynamics of the field changes. The field dynamics could change due to such events as wind direction change and sensor failures.

1. Introduction and background

Consider an area of surveillance and a network of remotely controlled sensors to cover its perimeter and inside. The perimeter sensors are utilized to initially detect the intrusion of any adversarial target into the field. Any target detected as such is then reported to a command and control authority (e.g., a higher level supervisory control) for further actions. In some applications, further tracking of the targets inside the field is required and is done using additional sensors within the field. In [2] and [3], targets within the field are tracked using a combination of two or three sensors – a tracker. The dynamic selection of the trackers as different targets move within the field then becomes an optimal control problem, usually with the dual objective of minimizing the tracking error and power consumption by the sensors. This optimal control problem is solved every time that a sensor fails, tracking error for a tracker exceeds some predefined threshold, or a new target intrudes into the field and is detected by one or more of the perimeter sensors. In security applications, the command and control action may include blocking of any intruders as soon as they are detected by the perimeter sensors.

In this article we will mainly focus on perimeter detection problem. We will assume that a number of remotely controlled sensors are already planted or distributed in the field, some of which to be used to cover the field's perimeter and others to be used for tracking targets inside the field. Since full coverage of the perimeter is essential for surveillance mission to succeed, the perimeter sensors need to be kept on continuously. Sensor cost, therefore, depends on, among other things, the amount of energy that they consume. The perimeter detection problem then reduces to optimally determining the subset of sensors in the field such that the overall cost is minimized and full coverage of the field's perimeter is insured. Obviously, any sensor not used for perimeter detection can be potentially used for target tracking inside the field.

The problem of perimeter/boundary detection in target tracking has been addressed by a number of researchers [1]. The main focus has been on the radar detection. Bonneau [4] considers the ground moving target radar detection problem. The objective is to consistently track targets moving through non-homogeneous regions such as forest and urban areas. Pyeron [5] describes a robust hierarchical probabilistic framework for visual target tracking. The author proposes a robust and efficient technique for detection of region boundaries in a sequence of images, which is the foundation for a hierarchical framework for visual target tracking. This robust technique is a multi-hypothesis test that searches for a boundary that maximizes a certain performance criterion. Deng [6] describes the target boundary problem, where a two-dimensional target tracking model is applied. Kalivas [7] presents an approach which models the image sequence including information about the position and shape of target. A genetic algorithm (GA) was used in [8] to solve the perimeter detection problem. The merit of using GA is that a relatively good result can be obtained in relatively short time. But it is well known that GA does not necessarily provide an optimal solution, and furthermore, the convergence and accuracy of the solution is very much dependent on the initial conditions set for the problem.

The rest of the paper is organized as follows: Section 2 gives a brief introduction on the adaptive control system applied in

* Corresponding author.
E-mail addresses: jafari@rci.rutgers.edu (M.A. Jafari), davood.golmohammadi@umb.edu (D. Golmohammadi).

0377-2217/$ - see front matter © 2008 Published by Elsevier B.V.
doi:10.1016/j.ejor.2008.06.018

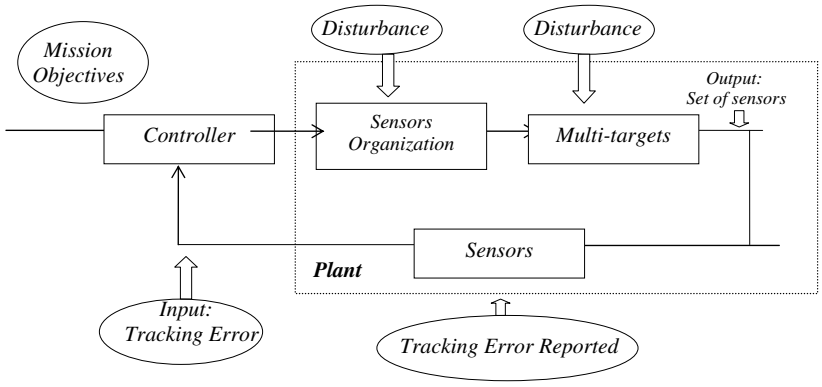


Fig. 1. Adaptive control schemes in multi-targets tracking.

precision target tracking. Section 3 introduces network flow formulation of the perimeter detection problem. In Section 4, extensions to polygons and general cases will be explained. Section 5 discusses optimization problem in the network flow model. Section 6 discusses supervisory field with insufficient sensors while Section 7 concludes this article.

2. A brief note on the adaptive supervisory control system

As briefly stated above, perimeter detection problem is often part of a larger control and monitoring problem and that the solution to this problem often changes depending on the dynamical changes in the system. Disturbances such as sensor failures, sensor errors, and changing weather conditions, which often results in changes of sensor field of view or range, are reasons to seek new optimal solutions. In this section we briefly describe a control and monitoring framework which includes perimeter detection as well as multi target tracking.

Fig. 1 provides an abstraction of this framework. Sensor organization routine decides which sensors are used for perimeter and which ones for target tracking. The box with "Multi targets" together with sensor organization defines the plant or process under surveillance or control. At each discrete time step the controller must perform, based on current system status, such functions as perimeter detection, new target detection, target tracking, tracker switching, and a variety of other computations. In its general form, multiple objectives (i.e., target detection, tracking, and the minimization of power usage) are considered. For the perimeter detection full coverage of the perimeter is the primary objective but certainly this must be achieved with minimum cost. If mission objectives are not met, i.e., an out-of-control situation arises, there may be a need for reorganization of sensors. Thus, the controller jointly with the sensor organization routine must ensure that the mission objectives are met on a continuous basis.

We note that the total power consumption is composed of two parts: the power consumed by the sets of perimeter sensors and by the tracker sensors. A sensor may be a member of more than one of these sets, i.e., it can be a perimeter and a tracking sensor with the former one being dominant. In other words, a perimeter sensor always remains on regardless of its tracking status, and as such one can decompose the overall problem of sensory optimization problem into two problems: optimal selection of perimeter sensors and optimal tracking. Here we will only focus on the former problem.

3. Network flow formulation of the perimeter detection problem

We will show that the perimeter sensory coverage problem can be formulated as a network flow model and thus be solved in a polynomial time. Depending on weather conditions, geometric constraints, command and control mission(s), the perimeter of the supervisory field can change dynamically within quite short amount of time. Furthermore, in order to fully cover the perimeter a large number of sensors may be required. Thus, it is of utmost importance to develop an algorithm, which is computationally of polynomial complexity and can be used in an online basis with quasi-real time response.

In this article, we consider targets as single point physical bodies, thus the target's shape is ignored. This is a reasonable assumption in lieu of the fact that the surveillance field is usually very large. We will also assume that a given perimeter is composed of mutually exclusive segments. Our formulation, however, will not be limited to any specific geometric shapes.

In what follows, we will first present the optimization problem in general terms and discuss its complexity. We will then show that it can be formulated as a network flow problem. We will first focus on a single segment and then expand the solution to closed boundaries. We will provide illustrative examples as part of our presentation.

3.1. Perimeter detection optimization problem

Let us assume that there are a total of n sensors, each with its own operational cost which includes the cost of power usage at some remote location within the field. Other costs, such as deployment cost, can also be included in our formulation. The problem we consider here is to select from this set a subset of sensors which fully covers the perimeter of the field with minimum total cost. We assume that every one of these n sensors can be used as a perimeter sensor and can be also used for target tracking (either simultaneously or separately). The following assumptions will be made for our formulation:

Assumption 1. The possible geometric forms include single segment; combination of several segments; and polygons.

Assumption 2. Although the monitoring scope of a sensor is circular, windy conditions render non-circular scopes, with decreased range upwind and extended range downwind. We will initially assume the circular cases. We will denote by C_i, $i = 1,2,\ldots,n$, the circular monitoring range of sensor i. We will discuss the elliptical case later, and will show that our solution methodology remains applicable.

Assumption 3. The supervisory field S is considered as a two-dimensional plane, and its perimeter is a polygon or a set of connected line segments. We will denote by Per the perimeter of the field.

Assumption 4. Sensors can be different in their hardware sensitivity to wind, power usage, and power sources. As long as the sensors maintain convex shaped detection scope, our network flow transformation algorithm will remain applicable.

The perimeter detection problem can now be stated as

$$\text{Minimize total cost} = \sum \text{Cost}_i \cdot \delta(i)$$
$$\text{s.t. } \delta(i) = 0 \text{ or } 1$$
$$\forall \text{ points } P \in \text{Per} \rightarrow P \in \cup C_i * \delta(i) \quad i = 1, 2, \ldots, n.$$

This problem is basically a 0–1 integer programing problem where total cost constitutes the objective function and constraints are defined on full coverage of the field. At the first glance, one may find this problem similar to classical problems, such as Vertex-Cover Problem, Minimum Cover Problem, Squares-Cover Problem, or Disks-Cover Problem [9]. It is well known that these problems are NP complete problems. As we will see shortly, however, we can reduce our problem into a network flow problem and solve it in polynomial time.

In the following sections, we will present the network flow formulation of the above optimization problem. We will start with a single segment first (i.e., *Per* will reduce into a single line) followed by full polygons.

3.2. Formulate single segment cover problem as network flow model

To transform the single segment problem into network flow model, we define a directed graph $G = (N, A)$ consisting of a set of N nodes and a set A of arcs whose elements are ordered pairs of distinct nodes [10]. A directed network is a directed graph whose nodes and/or arcs have associated numerical values (e.g., cost or capacity.). Here, we define the arc value as C_{ij} for arc (i, j). Further, N denotes the number of nodes and $m = |A|$ denotes the number of arcs in G. We will show by construction that the single segment cover problem can be reduced to a network flow model.

We begin by defining some basic elements in the network:

Source and sink: For segment AB, denote A as the starting point, and B as the ending point. A is regarded as the source of the network, and B as the sink of the network.

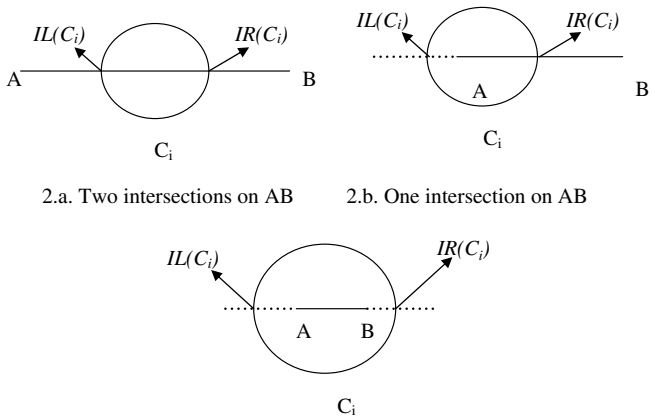
Intersecting points: Consider the set C of circles that intersects the segment AB. Each circle C_i intersecting with AB is represented by a node i in the network. For circle C_i, we define $IL(C_i)$, $IR(C_i)$ two intersection points along segment AB as we move from left to right. If C_i has only one intersecting point with AB, then, we have: $IL(C_i) = IR(C_i)$. In the case of no intersection points, these two points are undefined.

If both $IL(C_i)$ and $IR(C_i)$ located between A and B, then point closer to A is $IL(C_i)$ and the point closer to B is $IR(C_i)$. For points $IL(C_i)$ and $IR(C_i)$, if there is only one point located between A and B, but another is located outside AB (should be in the extension of AB *or* BA), then we will define the outside point as $IL(C_i)$ if it is on the left hand side of A, and $IR(C_i)$ if it is on the right hand side of B. Fig. 2a–c illustrate these cases.

Distance: Define $d(IL(C_i))$ as a distance measure between point A and $IL(C_i)$ along segment AB. Similarly $d(IR(C_i))$ is a distance measure between point A *and*, $IR(C_i)$. To calculate these measures, consider points A (X_A, Y_A), B (X_B, Y_B), $IL(C_i)(X_{Li}, Y_{Li})$, $IR(C_i)(X_{Ri}, Y_{Ri})$. We have

$$d(IL(C_i)) = \begin{cases} \sqrt{(X_A - X_{Li})^2 + (Y_A - Y_{Li})^2} & \text{if } IL(C_i) \text{ is between A and B} \\ -\sqrt{(X_A - X_{Li})^2 + (Y_A - Y_{Li})^2} & \text{otherwise} \end{cases}$$

$$d(IR(C_i)) = \begin{cases} \sqrt{(X_A - X_{Ri})^2 + (Y_A - Y_{Ri})^2} & \text{if } IR(C_i) \text{ is between A and B} \\ -\sqrt{(X_A - X_{Ri})^2 + (Y_A - Y_{Ri})^2} & \text{otherwise} \end{cases}.$$



2.a. Two intersections on AB 2.b. One intersection on AB

2.c. Ci covers the whole segment, intersections are outside.

Fig. 2. Various position of $IL(C_i)$ and $IR(C_i)$.

In this paper, distance is assumed to be the real value.

There are some special cases:

- For one intersection point we have $d(IL(C_i)) = d(IR(C_i))$.
- For circle C_i $d(IL(C_i)) \leqslant 0$ and $d(IR(C_i)) \leqslant 0$ mean that $IL(C_i)$ and $IR(C_i)$ are located on the extension of BA and AB, respectively. C_i can cover both A and B, so it covers the whole segment AB as shown in Fig. 2c. In this case, C_i may be one of our final optimal solutions.

The following construction steps convert our problem into a network flow optimization problem. The construction of the network is straightforward – sensors or circles are designated as nodes in the network. Any two circles which have common intersections along the segment will have arc joining them.

Network flow construction model

Step 1: Construct network nodes: First, consider those circles (sensors) with negative values for $d(IL(C_i))$, $i = 1, 2, \ldots, n$. Relabel these circles $C_1, C_2, \ldots, C_k$ in ascending order of $d(IR(\cdot))$. Second, consider the circles with positive $d(IL(C_i))$. Relabel these circles starting with C_{k+1} and continue with C_{k+2}, and so on, in ascending order of $d(IL(.))$. This continues until the circle with greatest $d(IL(C_i))$ is labeled C_n. These labels will be used in the network. Fig. 3 illustrates relabeling of circles according to their intersection with segment AB.

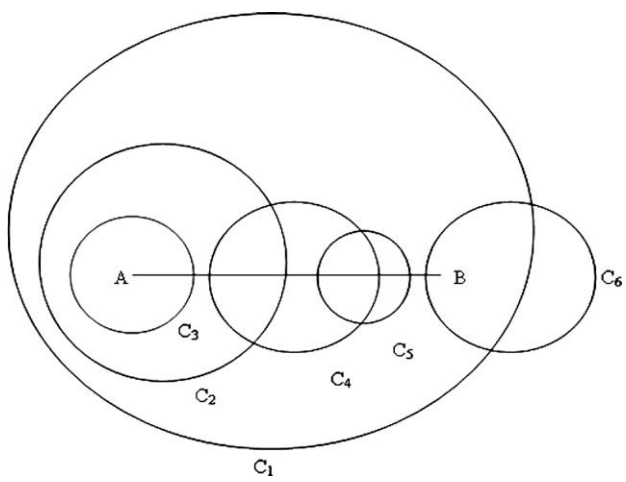


Fig. 3. Relabeling of sensors according to their intersections with segment AB.

Step 2: Construct network arcs:
Connection to source node: Circle C_i with negative $d(IL(C_i))$ is connected to the source node by *arc* A_{si} where the source is the tail and C_i is the head. Also C_i is connected to C_j by *arc* A_{ij} if $d(IL(C_i)) \leqslant 0$ and $d(IL(C_j)) \leqslant 0$.
Connection to Sink node: Circle C_i with negative $d(IR(C_i))$ is connected to the sink node by *arc* A_{is} where C_i is the tail and the sink is the head. C_i is connected to C_j by *arc* A_{ij} if $d(IR(C_i)) \leqslant 0$ and $d(IR(C_j)) \leqslant 0$.
Connection between intermediate nodes: Check n circles one by one in the following order $C_1, C_2, C_3 \ldots, C_n$. For circle C_i with positive $d(IL(C_i))$, and circle C_j satisfying

$$|d(IR(C_i))| \geqslant d(IL(C_j)) \geqslant d(IL(C_i))$$

or

$$|d(IR(C_i))| \geqslant |d(IR(C_j))| \geqslant d(IL(C_i)) \quad i \neq j$$

add *arc* A_{ij} between C_i to C_j.

- If both $d(IL(C_i))$ and $d(IL(C_j))$ are positive, the circle with smaller $d(IL(\cdot))$ is the tail node of A_{ij}, and the circle with larger $d(IL(\cdot))$ is the head node of A_{ij}.
- If $d(IL(C_i))$ or $d(IL(C_j))$ is negative, then the circle with negative $d(IL(\cdot))$ is the tail node of A_{ij}, and the circle with positive $d(IL(\cdot))$ is the head node of A_{ij}.
- If both $d(IL(C_i))$ and $d(IL(C_j))$ are negative and both $d(IR(C_i))$ and $d(IR(C_j))$ are positive, the circle with smaller $d(IL(.))$ is the tail node of A_{ij} and the circle with larger $d(IL(C_i))$ is the head node of A_{ij}.
- If for some circle C_i, both $d(IL(C_i))$ and $d(IR(C_i))$ are negative, that means we have found the optimal solution and we do not need to construct the network, and the algorithm terminates.

In summary, only when two circles have common interval intersections in the segment, can an arc between these two circles in the network exist. In addition, to reduce the running time of the construction process, if the relationship between C_i and C_j has been ascertained, we will not examine them again.

3.3. An illustrative example

Consider Fig. 4 with 10 circles $C_1, C_2 \ldots, C_{10}$ and line segment *AB*.
The detail steps to build the network flow model for the above case is given in Appendix A. The network flow model is shown in Fig. 5.

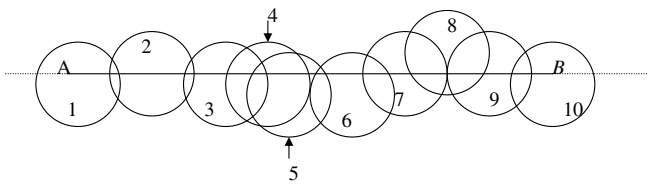


Fig. 4. A line segment and ten intersecting circles.

4. Extensions to polygons and general cases

In this section, we will present the extension of the network flow construction model to polygons, sparse sensor distribution, and to sensors with ellipsoidal ranges.

4.1. Polygon

We will assume that a polygon is made of a set of line segments L_j, $j = 1,2,\ldots$ The polygon can be of any regular or irregular shape as long as it is defined as a set of connected line segments, close ended or open ended.

Notation:

V: a point in polygon to identify the start and finish of the construction

$$g(C_i; P) = \begin{cases} 1 & \text{if } P \text{ is inside } C_i \text{ or on } C_i \\ 0 & \text{otherwise} \end{cases}.$$

$IS(C_i)_j$: Intersection point between C_i and L_j, $j = 1,2,\ldots$
$d(IS(C_i)_j)$: Distance between V and $IS(C_i)_j$ along the perimeter.

We begin our construction from V, which is regarded as the source of the network and move along L_j, $j = 1,2,\ldots$ in a clockwise direction along the perimeter starting from V. After searching all of C_is, the construction ends at V.

Network flow construction model – general case
Step 1: For each C_i, $i = 1,2, \ldots, n$, compute $IS(C_i)_j$ and $d(IS(C_i)_j)$. If C_i has more than one intersection along the perimeter, choose the greatest distance value for it. Relabel these circles C_i, $i = 1,2, \ldots, n$ according to the ascending order of their distance measures. This is min–max operation $\min(\max(d(IS(C_i)_j)))$ where max operation is on each circle and min operation is between circles.
Step 2: For each C_i, $i = 1,2, \ldots, n$, compute $g(C_i;V)$. C_i with nonzero $g(C_i;V)$ is defined to be connected to the source V by *arc* A_{vi} where the source is the tail and C_i is the head. $P = \{C_i, i = 1,2,\ldots,n \mid g(C_i;V) = 1\}$.
Step 3: To determine whether C_i intersects another circle C_k along L_j, $j = 1,2, \ldots$, compute $g(C_k;IS(C_i)_j)$, $j = 1,2, \ldots$ If $(C_k;IS(C_i)_j) > 0$, C_i intersects C_k. An *arc* A_{ik} is then added to the network, where the circle with smaller $\max(d(IS(.)_j))$ is assigned as the tail, while the circle with greater $\max(d(IS(\cdot)_j))$ is assigned as the head.
Step 4: For each circle C_i in P, *arc* A_{iv} *is defined connecting* C_i to the sink node.

Consider the example shown in Fig. 6.
The constructed network model is shown in Fig. 7.

4.2. Sparse sensor distribution area – the case of areas with holes

Sparse sensor distribution can often result in regions that have no sensor coverage, or 'holes'. Fig. 8 below provides one such example. Generally speaking, such a field can be regarded as a polygon, and the above algorithm could still be applied. In cases where the hole perimeter is extremely complicated with small hole sizes, we could do some normalization before applying the transformation algorithm. The purpose of normalization is to simplify

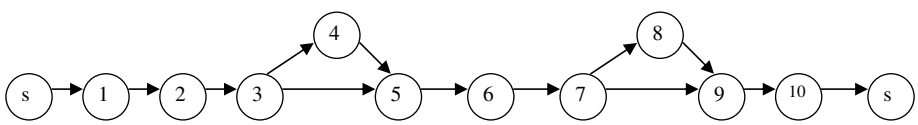


Fig. 5. The network flow model for the illustrative example.

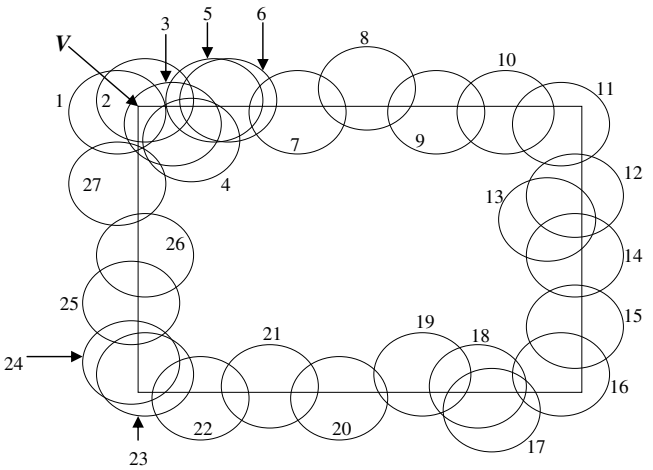


Fig. 6. A Polygon shaped perimeter detection example.

without compromising the unmonitored area. In Fig. 8, sides *A-B-C-D-E-F-G* is replaced by a simple side AG.

4.3. Sensors with ellipsoidal range

Thus far we have been working with sensors which have circular range. In practice, windy conditions render the elliptical scope shown in Fig. 9, with decreased range unwind and extended range downwind.

Windy conditions not only change the supervisory scope of a sensor from a circle to an ellipse, but also may produce some certain rotated angle on the ellipse as shown above. Therefore, to obtain the detailed properties of the sensor scope under the windy conditions, we need to know the long range, short range and the rotation angle of the ellipse under some certain wind conditions.

To construct the network flow model, we only considered the relationship between interconnected circles and between circles and segments along the perimeter. We did not account for the detailed properties of the circles, such as its position, radius, etc. We also note the following points: (1) both circles and ellipses are convex sets. Thus, when two circles or ellipses intersect with each other as shown in Fig. 10, the intersecting points are arranged and distributed similarly, except that in the case of ellipses a rotated angle exists. (2) Also due to convexity, circles and ellipses intersect with a line segment in a similar manner.

Thus the construction model described above can also be applied to sensors with ellipsoidal ranges without any changes.

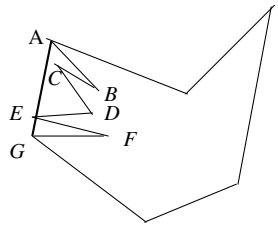


Fig. 8. Sparse Sensor Distribution Area (Hole)—Normalization.

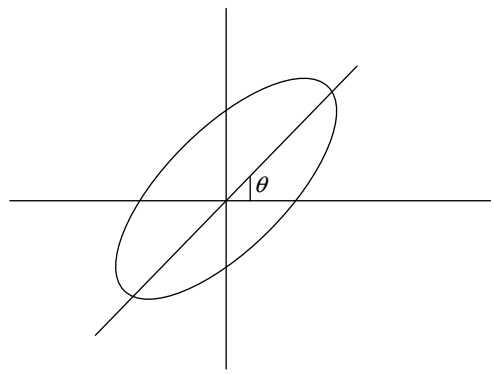


Fig. 9. Sensor Scope Range—Ellipse Case.

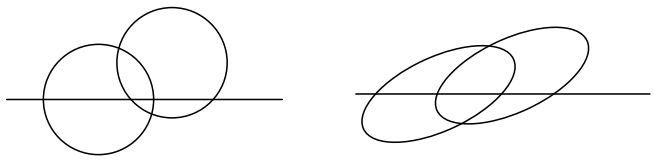


Fig. 10. Comparison of Circles and Ellipses.

5. Optimization problem in the network flow model

Let's go back to our original optimization problem – an optimal solution to this problem will give us minimum cost set of sensors that can fully cover the perimeter. Of course it is always possible not to have a feasible solution. We will discuss this later in this article. For now let us assume that there is a feasible solution. For the transformed model *G*, this feasible solution will correspond to a path from the source node to the sink node. Each node in such a path corresponds to a sensor circle which intersects with the

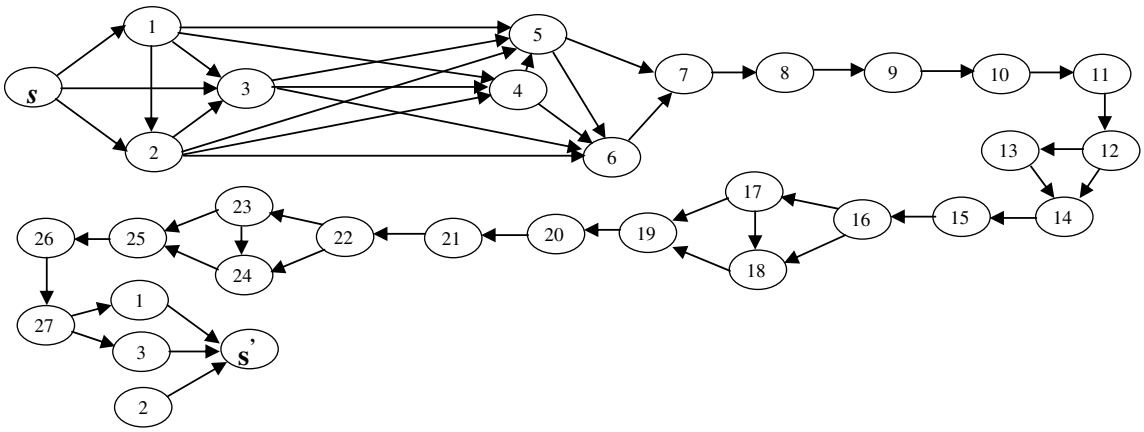


Fig. 7. The constructed network flow model for Fig. 6.

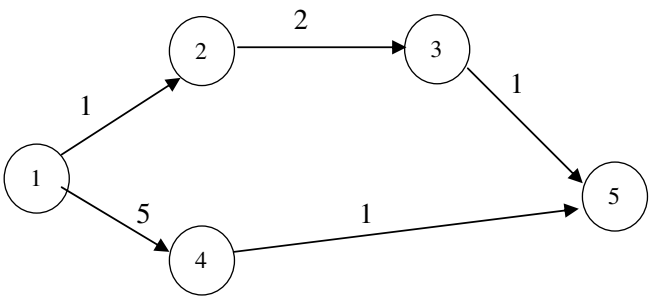


Fig. 11. Network arc cost—non-unit case.

perimeter and covers part of it. And since an arc exists only between intersecting circles, any subset of connected nodes in the path should be equal to a group of circles that can fully cover a segment of the perimeter. In the case of unit cost for all sensors the shortest path will have the least number of nodes in G. This will yield a solution for the original problem with the smallest number of sensors which cover the perimeter. In the case that sensors are not the same and have non-unit cost, the optimal solution for the original problem is still the shortest path from source to sink in G. To better understand this, consider the following network shown in Fig. 11.

Arc weights in the above example are the cost of sensors. For instance, the arc cost between node 1 and node 4 is 5 reflecting the cost of sensor 4. For arcs (3, 5) and (4, 5) cost factor is 1 since both arcs refer to using sensor 5. The problem of defining these weight factors is outside of the scope of this article. For the above problem, the shortest path from node 1 to node 5 is {(1,2);(2,3);(3,5)}.

For the network model of Fig. 7, assuming unit cost for all sensors, an optimal solution is {3, 4, 6, 7, 8, 9,10, 11, 12, 14, 15, 16, 17, 19, 20, 21, 22, 24, 25, 26, 27}. Obviously, different senor costs will yield different optimal solutions.

5.1. Computational complexity

The shortest path problem is a classical problem in network flow analysis. There are a number of methods, all with complexity which is less than or equal to $O(n^2)$, such as Dijkstra's algorithm. On the other hand, the network flow construction algorithm described above has a complexity of $O(n^2)$. Thus, the overall optimization problem can be solved in a polynomial time.

6. Supervisory field with insufficient sensors

If the number of available sensors is insufficient to fully cover the surveillance field, the field perimeter can be adjusted and reduced to one such that the new enclosed area is fully covered by the existing sensor distribution. Fig. 12 shows one such field reduction example. To apply our construction model, the non-linear perimeter can be approximated by linear segments.

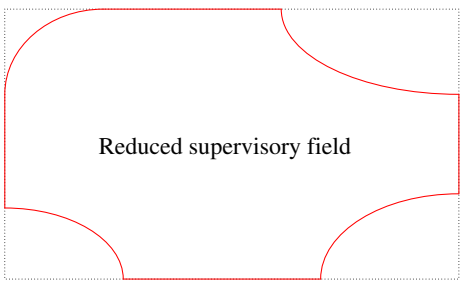


Fig. 12. Reduced Surveillance Field (insufficient sensors case).

7. Conclusion

In this article, the perimeter detection optimization problem in field surveillance and target tracking is discussed. It was shown that the problem of optimal sensory selection can be reduced to a network flow problem and can then be solved using any existing classical methodology. Sensors are associated with a cost factor reflecting their operational characteristics and power usage. The detection scope of the sensors are assumed to be convex. The overall complexity of the problem reduces to $O(n^2)$ where n is the number of sensors.

Appendix A. The construction of the network model proceeds as follows

Notation

G: un-labeld list of circles, $G = \{C_i \mid i = 1,2,\ldots,n\}$ at the start of construction, $G = Null$ after the construction is finished.

F: floating list of circles, $F = Null$ at the start of construction, $F = Null$ after the construction is finished.

K: constructed list of circles; $K = Null$ at the start of construction, $K = \{C_i \mid i = 1,2,\ldots,n\}$ after the construction is finished.

The network flow model for the Segment-Circle Cover problem is constructed by the following steps:

Node list = {1, 2, 3, 4, 5, 6, 7, 8, 9,10}; $G = \{C_1, C_2, \ldots, C_{10}\}$.

Perform the following steps based on the algorithm described earlier:

Step 1: $G = \{C_2,\ldots,C_{10}\}$; $F = \{C_1\}$; $K = Null$; *arc* A_{s1} is added;
Step 2: $G = \{C_3,\ldots,C_{10}\}$; $F = \{C_1, C_2\}$; $K = Null$; *arc* A_{12} is added;
Step 3: $G = \{C_3,\ldots,C_{10}\}$; $F = \{C_2\}$; $K = \{C_1\}$;
Step 4: $G = \{C_4,\ldots,C_{10}\}$; $F = \{C_2, C_3\}$; $K = \{C_1\}$; *arc* A_{23} is added;
Step 5: $G = \{C_4,\ldots,C_{10}\}$; $F = \{C_3\}$; $K = \{C_1, C_2\}$;
Step 6: $G = \{C_5,\ldots,C_{10}\}$; $F = \{C_3, C_4\}$; $K = \{C_1, C_2\}$; *arc* A_{34} is added;
Step 7: $G = \{C_6,\ldots,C_{10}\}$; $F = \{C_3, C_4,C_5\}$; $K = \{C_1, C_2\}$; *arc* A_{35}, A_{45} are added;
Step 8: $G = \{C_6,\ldots,C_{10}\}$; $F = \{C_4,C_5\}$; $K = \{C_1, C_2, C_3\}$;
Step 9: $G = \{C_6,\ldots,C_{10}\}$; $F = \{C_5\}$; $K = \{C_1, C_2, C_3, C_4\}$;
Step 10: $G = \{C_7,\ldots,C_{10}\}$; $F = \{C_5, C_6,\}$; $K = \{C_1, C_2, C_3, C_4\}$; *arc* A_{56} is added;
Step 11: $G = \{C_7,\ldots,C_{10}\}$; $F = \{C_6\}$; $K = \{C_1, C_2, C_3, C_4,C_5\}$;
Step 12: $G = \{C_8,\ldots,C_{10}\}$; $F = \{C_6,C_7\}$; $K = \{C_1, C_2, C_3, C_4, C_5\}$; *arc* A_{67} is added;
Step 13: $G = \{C_8,\ldots,C_{10}\}$; $F = \{C_7\}$; $K = \{C_1, C_2, C_3, C_4,C_5, C_6\}$;
Step 14: $G = \{C_9, C_{10}\}$; $F = \{C_7, C_8\}$; $K = \{C_1, C_2, C_3, C_4, C_5, C_6\}$; *arc* A_{78} is added;
Step 15: $G = \{C_{10}\}$; $F = \{C_7, C_8, C_9\}$; $K = \{C_1, C_2, C_3, C_4, C_5, C_6\}$; *arc* A_{79} and A_{89} are added;
Step 16: $G = \{C_{10}\}$; $F = \{C_8, C_9\}$; $K = \{C_1, C_2, C_3, C_4,C_5, C_6, C_7\}$;
Step 17: $G = \{C_{10}\}$; $F = \{C_9\}$; $K = \{C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8\}$;
Step 18: $G = Null$; $F = \{C_9, C_{10}\}$; $K = \{C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8\}$; *arc* A_{9-10} is added;
Step 19: $G = Null$; $F = \{C_{10}\}$; $K = \{C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8, C_9\}$;
Step 20: $G = Null$; $F = Null$; $K = \{C_1, C_2, C_3, C_4,C_5, C_6, C_7, C_8, C_9, C_{10}\}$; *arc* A_{10-s} is added.

References

[1] I. Kadar, Optimum geometry selection for sensor fusion, in: I. Kadar (Ed.), Signal Processing, Sensor Fusion and Target Recognition VII, SPIE-3374, The International Society for Optical Engineering, Bellingham, 1998, pp. 13–15.
[2] Richard A. Burne, Anna L. Buczak, Yaochu Jin, Vikram R. Jamalabad, Ivan Kadar, Eitan R. Eadan, A self-organizing, cooperative sensor network for remote surveillance: Currtent results, in: Proceedings of the SPIE, The International Society for Optical Engineering, 1999.
[3] Richard A. Burne, Ivan Kadar, J. Whitson, A.L. Buczak, A self-organizing, cooperative sensor network for remote surveillance: Improved target tracking

results, in: Proceedings of the SPIE, vol. 4232, The International Society for Optical Engineering, 2001, pp. 313–321.
[4] R.J. Bonneau, A spectral approach to image-enhanced moving target radar detection, in: Proceedings of the IEEE National Radar Conference, 2001, pp. 31–34.
[5] Pyeron, Michael, Waks, Amir, Gregoriou, Georghios, Tretiak, Oleh, Bar-Kana, Izhak, A robust hierarchical probabilistic framework for visual target tracking, in: Proceedings of the IEEE International Conference on Acoustics, vol. 4, Speech and Signal Processing, 1991, pp. 2457–2460.
[6] Deng, Keqiang, Wilson, Joseph, N., Approximation – based video tracking system, in: Proceedings of the SPIE, vol. 1568, USA, 1991, pp. 304–312.
[7] D.S. Kalivas, A.A. Sawchuk, R. Chellappa, Estimation and segmentation of image sequences, in: Proceedings of the SPIE, vol. 829, WA, USA, 1987, pp. 15–23.
[8] R.A. Burne, A.L. Buczak, V.R. Jamalabad, I. Kadar, E.R. Edan, A self-organizing, cooperative sensor network for remote surveillance, sensors, C3I, information, and tranining technologies for law enforcement, in: E.M. Carapezza, D.B. Law (Eds.), Proceedings of the SPIE, vol. 3577, The International Society for Optical Engineering, Bellingham, 1998, pp. 124–134.
[9] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, Introduction to Algorithms, second ed., 2001, pp. 971–1022.
[10] Ravindra K. Ahuja, Thomas L. Magnanti, James B. Orlin, Network Flows, Theory, Algorithms, and Applications, 1993, pp. 125–213.