NSF GRANT #0400140
NSF PROGRAM NAME: Operations Research

Minimax Sensor Location in the Plane

Tom M. Cavalier
The Pennsylvania State University
University Park, PA 16802

Enrique del Castillo
The Pennsylvania State University
University Park, PA 16802

Whitney A. Conner
The Pennsylvania State University
University Park, PA 16802

Abstract: This paper addresses the problem of locating a finite number of sensors to detect an event in a given planar region. The objective is to minimize the maximum probability of non-detection where the underlying region consists of a convex polygon. The sensor location problem has a multitude of applications, including the location of aircraft detection sensors, the placement of sentries along a border to detect enemy penetration, the detection of nuclear tests, and the detection of natural and hazardous man-made events. The problem is a difficult nonlinear nonconvex programming problem even in the case of two sensors. A fast heuristic based on Voronoi polygons is developed in this paper. The algorithm can quickly generate high-quality solutions. Computational experience is provided.

1. Introduction: Suppose that in a confined planar region an event occurs. The problem is to position a finite number of sensors so that the probability of detecting such an event is maximized. The effectiveness of each sensor decreases with its distance from the event. The solution to such a problem has numerous applications. For the military, the location of aircraft detection sensors, the placement of sentries along a border to detect enemy penetration, and the detection of nuclear tests are just three examples. The detection of natural events as well as the detection of hazardous man-made events are also important applications. These events can be the result of accidents as well as deliberate terrorist acts.

The above applications require a network of sensors in which it is crucial that an event does not go undetected because of its location within a region. This can be achieved with a minimax formulation because in such models, the poorest response is made to be as good as possible. Other examples in which a minimax model is appropriate include locating fire stations, hospitals and ambulance bases [Elzinga and Hearn 1972, Plastria 1995].

With this in mind, the problem can be thought of as a facility location problem with an objective of providing equitable service to the customers, where the facilities being located are the sensors, and the customers are the event (source) locations. In location theory terminology, a problem of this type can be classified as a continuous multifacility problem. However, in the problem studied here, the objective terms are not simply weighted distances as in most location problems, but a propagation function of distance converted into probabilities of detection.

It is assumed that events, within the scope of this research, produce a 'signal' that interacts with the environment through which they propagate. As the signal is transmitted from the event to the sensor (or vise-versa) various processes interact with the signal to produce noise. The causes for the noise can vary, depending on the type of event one wishes to detect, however regardless of the type of processes that produce the noise, the effect is a reduction in signal intensity, referred to as signal 'fading' or signal attenuation.

Throughout this paper the specific noise generated by the sensor itself, such as thermal noise from the electronics and circuits, or interference from other sensors will be neglected. That is, the idealized sensor will produce no noise and is perfectly shielded from interference.

Propagation models can be used to estimate and predict the fading of a signal produced by an event as it traverses through a region. In principle, the attenuation over every path between an event and sensor can be predicted. This method produces a propagation model based on ray-tracing or ray-paths. However, to make this possible, detailed knowledge on every terrain feature in the region is necessary, which leads to the computational effort being excessive for continuous regions [Saunders 1999]. Instead, a propagation model can be based on an empirical model, which is produced by fitting a function to an extensive set of actual path loss measurements. The model can then be used in environments similar to the original [Saunders 1999]. For many complex propagation models, no analytical form of the model exists. Rather, a function generator (i.e., black box), is used to generate a functional value based on various inputs.

Path loss can have various meanings, but here, it refers to the overall decrease in signal intensity due to an increase in distance between the event and the sensor. For example, for electromagnetic waves, as the signal radiates, the intensity 'concentration' spreads out in all directions into an approximate spherical surface [Perez 1998]. Thus, empirical models usually involve some form of an inverse function of distance, with the parameters being adjusted to match propagation conditions.

One such detection probability functions (dpf) is derived from the wireless communication industry, specifically macrocell placement. Macrocells are the towers used outdoors for providing mobile services with coverage from around 1 km to many tens of kilometers [Saunders, 1999]. The derived dpf, referred to as *power decay*, is given by

$\pi(d) = \dfrac{\alpha}{\mu + d^n}$, $0 < \alpha \le \mu$, $n > 0$ where $\pi(d)$ is the probability of detecting an event at a distance *d* away for a given ratio α/μ and exponent *n*. The probability of detecting an event at distance zero is given by the ratio α/μ. If $\alpha = \mu$, the detection probability is unity when the event occurs at a sensor. In addition, if *n* = 2 in this case, it becomes the standard inverse square law, which appears in models describing phenomena such as radio frequency intensity, gravitational attraction, and the force between charged particles.

A second dpf, termed *exponential decay*, is given by $\pi(d) = Ae^{-\beta d^n}$, $0 < A \le 1$, $\beta, n > 0$ where A is the probability of detection at $d = 0$ with parameter □ and exponent *n*. Exponential type models have had a reoccurring theme in many different disciplines for approximating the decay of a process. Exponential decay occurs whenever something changes at a rate proportional to itself. For example, in air quality management, a Gaussian model is used to predict the concentration levels of airborne pollutants from a source point. Pollutant concentration decreases exponentially in the Gaussian model.

A final dpf can be considered as a combination of the previous two dpf's and has the form $\pi(d) = 1 - e^{-k/d^n}$, $k, n > 0$. This has exponential form with a power decay (k/d^n) component and is referred to as *gravity decay*.

By changing the dfp parameters, families of detection probability functions can be generated. Also, the main objective here is to explore a solution procedure for producing good solutions to the sensor location problem in a timely fashion, for which these dpf's are sufficiently realistic. The algorithm developed in this paper is not tailored for a specific dpf, but addresses the problem in a more general sense.

The following assumptions are made for the sensor location problem (SELP).

1. The *m* sensors to be located are identical.
2. The region *S* where detection takes place is a convex polygon in the plane $(\mathbb{R}^2)$.
3. The event occurs with equal probability anywhere in the region *S*.
4. The region *S* contains no existing sensors.
5. The detection probability function is only dependent on the distance from the event to the *i*th sensor.
6. All sensors are in perfect working order with no internal or external influences affecting performance. This idealized sensor has perfect discrimination and no false detections.

That is, it will signal a detection only when the event is present.

7. The effectiveness of a sensor is independent of its location within the region *S*.
8. There are no constraints on the location of the sensors except that they must be located within the region *S*. That is, all points within the region *S* are candidates for sensor locations.
9. The probability of detection is a non-increasing, real-valued, continuous function of distance from the event.
10. The sensors operate independently of each other.

It is understood that these simplifications may not be valid in some actual cases. However, achieving solutions to simpler models can provide the foundation, insight and understanding to develop methods for more complex models. Because of the nonconvex nature of the objective function, verifying that the solution obtained is a global optimum can be quite difficult even for small problems.

Drezner and Wesolowsky [1997] addressed the problem of locating *p*-identical signal detectors on a unit line and a unit square. They considered two dpf's: $\pi(d) = \min\left\{1, R^{\lambda} / d^{\lambda}\right\}$ and $\pi(d) = e^{-kd}$. For the unit line problem, they used mathematical programming and a special algorithm designed to achieve the necessary conditions for optimality. While convergence of the algorithm was not proven, the algorithm always converged for their test problems and produced a better solution after each iteration.

For the square planar problem, four procedures were considered: a univariate search, a mathematical programming formulation, simulated annealing, and a Demjanov-type method. The planar problem was formulated using a finite number of possible event locations to produce a uniform grid covering the square region. This provides an approximate solution to the continuous problem.

The Demjanov-type algorithm, according to Drezner and Wesolowsky, provided the best solutions for the test problems in a 'reasonable' run time. The current station locations are improved by moving them in the direction of steepest descent. The direction is found using a method proposed by Demjanov [1968]. The objective function is then optimized along this direction using a one-dimensional optimization procedure.

From an application standpoint, research has been done in the field of wireless communication, specifically with respect to the base station location problem (BSTLP), a problem to which the SELP could be applied. With the increased demand for mobile communication services and deregulation acts (Tutschku 1998), the competition between service providers and potential revenues has driven the research for finding an optimal deployment of base stations which is both cost effective and provides the maximum possible coverage. The BSTLP involves locating multiple base stations within a region while providing an acceptable quality of service to mobiles (Howitt and Ham 1999). The BSTLP is different from the SELP in that the traffic and capacity on each base station is an important consideration.

For the problem of locating one sensor in a planar region, with the detection probability being a decreasing function of distance, the maximum probability will occur at the point(s) where the distance between the sensor and the event is a maximum, which corresponds to the event occurring somewhere on the boundary of the region. Thus the center of the smallest circle which encloses the region is the optimal placement for the sensor. This problem is commonly called a 1-center problem [Elzinga and Hearn 1972, Plastria 1995] in facility location. However, when the number of sensors $p > 1$, the problem is again a difficult nonlinear nonconvex programming problem.

For solving the 1-center problem, a method based on Voronoi diagrams [Shamos and Hoey, 1975] provides an understanding of the strengths of computational geometry. Many location problems, previously considered quite hard, are being addressed using methods based on various types of Voronoi diagrams [Plastria, 1995, pg 262; see for example Iri, *et al*., 1984, Okabe *et al*., 1992 and Aurenhammer, 1991,]. A Voronoi diagram can be described as follows [Preparata and Shamos, 1985]: Let *S* be a set containing a finite number of points (e.g. sensors) in the plane. For each point $\mathbf{p}_i \in S$, the set of locations in the plane closer to $\mathbf{p}_i$ than any other point in *S* is an element of the Voronoi polygon of $\mathbf{p}_i$, denoted $V(i)$. The Voronoi diagram associated with *S* is V = $\{V(1),...,V(n)\}$, where *n* is the number of points

contained in *S*. Several algorithms exist for computing Voronoi diagrams. Okabe *et al*. [1992] give a detailed description of incremental, divide-and-conquer, and plane-sweep methods.

Love *et al*. [1973] reformulated the continuous minimax (i.e., *p*-center) location problem as a constrained nonlinear programming problem and then used a penalty function method referred to as sequential unconstrained minimization technique (SUMT). Elzinga *et al*. [1976] used a technique in nonlinear programming known as Lagrangian duality to produce a problem which is easier to solve numerically. Drezner and Wesolowsky [1978] used an approach that involves the numerical integration of ordinary differential equations. Charalambous [1981] transforms the problem into a sequence of unconstrained squared Euclidean minisum problems which have analytical solutions. Brady *et al*. [1983] developed an interactive graphical program for addressing the minimax location of multiple facilities. The algorithms success depends on an individual's ability to perform a pattern recognition task. Because the minimax location problem is not everywhere differentiable subgradient algorithms [e.g., Chatelon *et al*., 1982; Demjanov 1968] have also been used. The *p*-center problem in an area was addressed by Suzuki and Drezner [1996], who developed a location-allocation type algorithm utilizing Voronoi polygons.

Although there are similarities between the *p*-center problem and the sensor location problem discussed in this paper, there are significant differences. The *p*-center minimizes the maximum distance between facilities and demand points. It is an allocation problem, and thus each demand point is assigned to and interacts with its closest facility. On the other hand, the objective function in the sensor location problem involves the product of probabilities, making each event point interact with <u>all</u> sensors. It is easy to see that an optimal set of facility locations for the *p*-center problem will, in general, be a very poor solution to the sensor location problem.

Whereas prior techniques for solving signal detection problems have relied on discretization, simulated annealing, and other search techniques, this paper attempts to exploit the geometry of the problem rather than consider the specific form of the objective function. This is done by deriving local information from Voronoi polygons and adjusting sensor positions by a technique called toward the largest peak (TLP). The remainder of this paper is organized as follows: The succeeding section provides a mathematical formulation of the problem. This is followed by a detailed description of the TLP algorithm, computational experience, and concluding remarks.

2. SELP Model: Suppose that an event occurs at location $\mathbf{z} = (x, y) \in S \subset \mathbb{R}^2$ and that the placement of *m* sensors (detectors) must be decided within region *S*. For convenience, an event occurring at location **z** will be referred to simply as event **z**. The probability of detecting the event with the *i*th sensor (located at $\mathbf{x}_i = (x_i, y_i)$) is $\pi(d_i)$ where $d_i = d(\mathbf{z}, \mathbf{x}_i) = \sqrt{(x - x_i)^2 + (y - y_i)^2}$ represents the Euclidean distance between sensor *i* and event **z**. The poorest response for the SELP will occur where the probability of non-detection is largest. Under the minimax criterion the largest probability of non-detection for the region should be made as small as possible. Let $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m)$ denote the vector of all sensor locations. Then if the *m* sensors operate independently of each other, the probability $g(\mathbf{z}, \mathbf{X})$ that an event **z** is not detected for a fixed set **X** of sensors is the product that each sensor individually does not detect event **z** and is given by:

$$g(\mathbf{z}, \mathbf{X}) = \prod_{i=1}^{m} \left[1 - \pi\left(d(\mathbf{z}, \mathbf{x}_i)\right)\right] \qquad (1)$$

Consider that an event occurring at $\mathbf{z} = (x, y)$ is a random variable with $p(\bullet)$ being a given probability density function (pdf). For any set $A \subseteq S$

$$\Pr\{\mathbf{z} \in A\} = \int_A p(a)da \text{ where}$$

$$\int_S p(a)da = 1,\ p(a) \geq 0 \quad \text{for all } a \in S \qquad (2)$$

The maximum value of the probability of non-detection $g(\mathbf{z}, \mathbf{X})$ is denoted by

$$\begin{aligned} \psi(\mathbf{X}) &= \max_{\mathbf{z}} p(\mathbf{z}) g(\mathbf{z}, \mathbf{X}) \\ &= \max_{\mathbf{z}} \left\{ p(\mathbf{z}) \prod_{i=1}^{m} \left[1 - \pi_i\left(d(\mathbf{z}, \mathbf{x}_i)\right)\right] \right\}. \end{aligned} \qquad (3)$$

The event pdf acts like a weight which produces a balance between the point of maximum probability of non-detection and the 'point' where the event is most likely to occur. In the case where the event is assumed to occur with equal probability anywhere in the region

S (i.e., the case of complete spatial randomness), $p(\mathbf{z})$ becomes a constant and can be dropped from equation (3) without affecting the optimal solution. By minimizing ψ, the poorest response is made as small as possible. Therefore, the problem can be written as

(SELP)

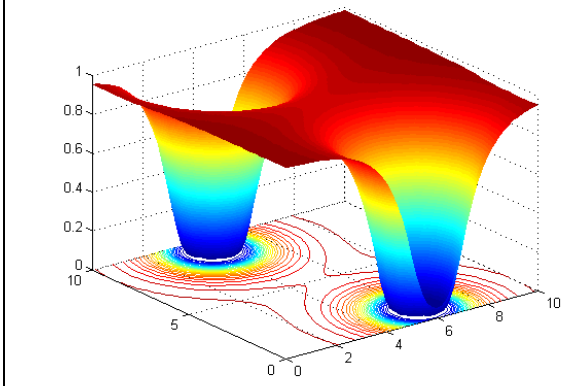
$$\min_{\mathbf{x}_1,\ldots,\mathbf{x}_m} \psi(\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_m) = \min_{\mathbf{x}_1,\ldots,\mathbf{x}_m} \max_{\mathbf{z}} \left\{ \prod_{i=1}^{m} \left[1-\pi_i\left(d(\mathbf{z},\mathbf{x}_i)\right)\right]\right\} \quad (4)$$

As mentioned earlier, (4) is a difficult nonlinear, nonconvex programming problem when $m>1$. A solution algorithm for (4) can be visualized as an iterative two-phase process. For a fixed set of sensors locations, the value of ψ is found (or approximated). In general, the calculation of the value of ψ is nontrivial and is often approached through discretization of the set *S*. The sensors are then moved and ψ is recalculated. As this process continues the goal is to decrease the value of ψ until no improvement can be achieved by sensor relocation. This is the basic strategy adopted in the TLP approach, which is outlined in detail in the following section.

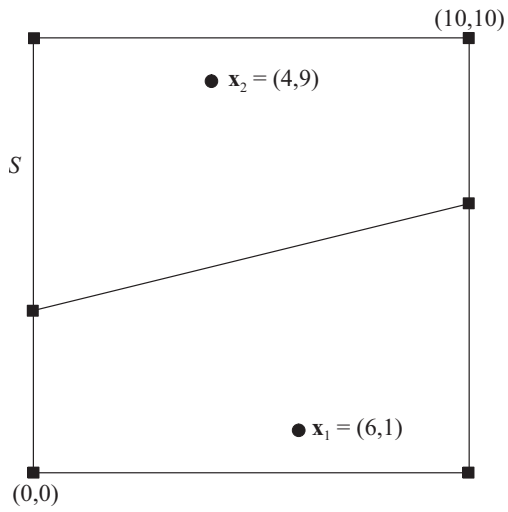
3. Toward the Largest Peak (TLP): Given a single sensor located in a convex polygonal region $S \subset \mathbb{R}^2$, the peaks (or points of maximum probability of non-detection ψ) occur at the vertices of *S* where the distance between the sensor and the event is a maximum. Now consider the case of two sensors where the set *S* is a square with vertices $(0,0)$, $(10,0)$, $(0,10)$, and $(10,10)$. Note from Figure 1(a) that the probability of non-detection is zero at the sensor locations with the peaks occurring along the boundary of the region S and the ridge running between the two sensors. This ridge corresponds precisely to the boundary of the Voronoi polygon containing each sensor (see Figure 1(b)). Thus, in this case, once the Voronoi polygons are determined the value of ψ can be found by determining the probability of non-detection of each of the vertices of the Voronoi polygons. This is more efficient that discretizing the entire set *S*, and is the basic strategy employed in the TLP algorithm to find an approximate value of ψ in the general *m*-sensor case.

Given *m* sensor locations, $\mathbf{x}_1,\ldots,\mathbf{x}_m \in S$, the algorithm determines the Voronoi polygons based on the points $\mathbf{x}_1,\ldots,\mathbf{x}_m$ and the boundary of the convex polygon *S*. As mentioned earlier, there are a number of algorithms for efficiently finding the Voronoi diagram of a finite set of points in $\mathbb{R}^2$ [see, for example, Preparata and Shamos, 1985 , Okabe *et al*., 1992]. [Okabe *et al*., 1992] also demonstrate how to define a supplemental set of points to avoid infinite Voronoi polygons. The intersection of the Voronoi diagram with the enclosing convex polygon *S* yields the desired Voronoi polygons. In the implementation that follows, the TLP algorithm generates the vertices of each Voronoi polygon using an efficient simplex-type procedure that takes into account the Voronoi diagram and the enclosing polygon *S*.

Once the probability of non-detection is determined for each Voronoi vertex, the sensor location within each Voronoi polygon is repositioned by moving it toward the largest local peak(s), thus reducing the peak (probability) and improving the overall objective value. Moving an individual sensor may not improve the objective at all; in fact, it may actually make the objective worse. Therefore, all sensors are moved simultaneously using local information derived from the individual Voronoi polygons. After repeating this process until no further improvement is possible, the sensors are repositioned using global Voronoi information to fine tune the solution. This second phase can result in a significant improvement in the final solution, however using only global information is generally inferior to using local information followed by global information. A more precise mathematical statement of the algorithm follows.



(a) Probability of Non-detection



(b) Voronoi Diagram

Figure 1. Two Sensors when *S* is a Square.

TLP Algorithm

PHASE I – This phase uses local Voronoi information to reposition the sensors.

Step 1. Set the iteration counter $k = 1$ and choose an initial set of sensors location $\mathbf{x}_1^k, ..., \mathbf{x}_m^k \in S$

Step 2. Find the Voronoi diagram corresponding to the discrete set of points $\mathbf{x}_1^k, ..., \mathbf{x}_m^k$ and form a tessellation of the set *S*. Let V_i^k represent the set of vertices of the Voronoi polygon corresponding to sensor location $\mathbf{x}_i^k$ (see Figure 2).

Step 3. Determine the probability of non-detection $q_j = q(\mathbf{v}_j) = \prod_{i=1}^{m}\left[1 - \pi_i\left(d\left(\mathbf{v}_j, \mathbf{x}_i^k\right)\right)\right]$ for each Voronoi vertex $\mathbf{v}_j \in \bigcup_{i=1}^{m} V_i^k$.

Step 4. For each sensor location $\mathbf{x}_i^k$, let $q_{i\max}^k = \max_{\mathbf{v}_j \in V_i^k} q_j$ and let $V_{i\max}^k = \left\{\mathbf{v}_j \in V_i^k : q_j = q_{i\max}^k\right\}$. That is, $q_{i\max}^k$ is the maximum probability associated with the vertices of the Voronoi polygon containing sensor location $\mathbf{x}_i^k$ and $V_{i\max}^k$ is the corresponding set of vertices. The current maximum probability of non-detection is $q_{\max}^k = \max_{i=1,..,m} q_{i\max}^k$.

Step 5. Compute $\mathbf{h}_i = \sum_{\mathbf{v}_j \in V_{i\max}^k} (\mathbf{v}_j - \mathbf{x}_i^k)$ and let $\mathbf{h}^t = (\mathbf{h}_1^t, ..., \mathbf{h}_m^t)$. Then $\mathbf{h} / \|\mathbf{h}\|$ is a unit direction vector that attempts to move each sensor location $\mathbf{x}_i^k$ toward the largest local peak(s) in its Voronoi polygon.

Step 6. For each sensor location $\mathbf{x}_i^k$, compute $\mathbf{x}_i^{k+1} = \mathbf{x}_i^k + \Delta\left(\mathbf{h}_i / \|\mathbf{h}\|\right)$ for step length Δ. Find V_i^{k+1}, the set of vertices of the Voronoi polygon corresponding to each location $\mathbf{x}_i^{k+1}$. Determine the probability of non-detection q_j for each point $\mathbf{v}_j \in \bigcup_{i=1}^{m} V_i^{k+1}$ and the corresponding value of $q_{\max}^{k+1}$.

Step 7. If $q_{\max}^{k+1} < q_{\max}^k$, then replace k by $k+1$ and go to Step 4. Otherwise, replace Δ by 0.75Δ. If $\Delta < \varepsilon$, then stop; otherwise, go to Step 6.

PHASE II – This phase uses global Voronoi information to reposition the sensor locations of Phase I.

Step 8. Given the set of sensors location $\mathbf{x}_1^k,...,\mathbf{x}_m^k \in S$ determined in Phase I, set the iteration counter $k=1$.

Step 9. Find the Voronoi diagram corresponding to the discrete set of points $\mathbf{x}_1^k,...,\mathbf{x}_m^k$ and form a tessellation of the set *S*. Let V_i^k represent the set of vertices of the Voronoi polygon corresponding to sensor location $\mathbf{x}_i^k$.

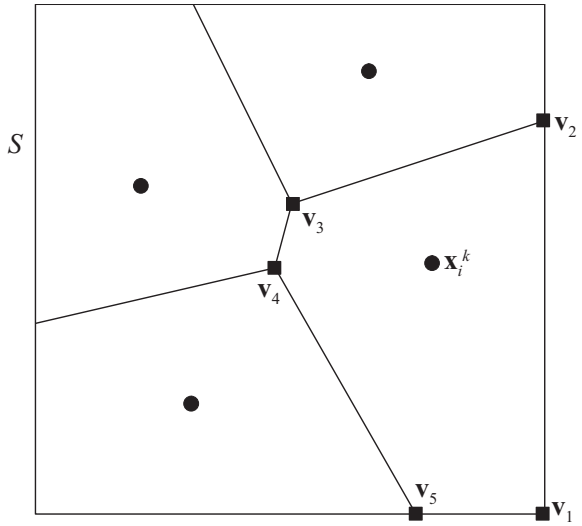
Step 10. Determine the probability of non-detection $q_j = q(\mathbf{v}_j) = \prod_{i=1}^{m}\left[1-\pi_i\left(d\left(\mathbf{v}_j,\mathbf{x}_i^k\right)\right)\right]$ for each Voronoi vertex $\mathbf{v}_j \in \bigcup_{i=1}^{m} V_i^k$.

Step 11. Let $q_{\max}^k = \max_{\mathbf{v}_j \in \bigcup_{i=1}^{m} V_i^k} q_j$ and let $V_{\max}^k = \left\{\mathbf{v}_j \in \bigcup_{i=1}^{m} V_i^k : q_j = q_{\max}^k\right\}$. That is, $q_{\max}^k$ is the maximum probability associated with the vertices of all Voronoi polygons and $V_{\max}^k$ is the corresponding vertex set. Let $V_{i\max}^k$ be the subset of $V_{\max}^k$ consisting of the points that are closest to $\mathbf{x}_i^k$.

Step 12. Compute $\mathbf{h}_i = \sum_{\mathbf{v}_j \in V_{\max}^k} (\mathbf{v}_j - \mathbf{x}_i^k)$ and let $\mathbf{h}^t = (\mathbf{h}_1^t,...,\mathbf{h}_m^t)$. In this case, $\mathbf{h}/\|\mathbf{h}\|$ is a unit direction vector that attempts to move sensor location $\mathbf{x}_i^k$ toward the closest global peak(s).

Step 13. For each sensor location $\mathbf{x}_i^k$, compute $\mathbf{x}_i^{k+1} = \mathbf{x}_i^k + \Delta\left(\mathbf{h}_i/\|\mathbf{h}\|\right)$ for step length Δ. Find V_i^{k+1}, the set of vertices of the Voronoi polygon corresponding to each location $\mathbf{x}_i^{k+1}$. Determine the probability of non-detection q_j for each point $\mathbf{v}_j \in \bigcup_{i=1}^{m} V_i^{k+1}$ along with $q_{\max}^{k+1}$, $V_{\max}^{k+1}$, and $V_{i\max}^{k+1}$.

Step 14. If $q_{\max}^{k+1} < q_{\max}^k$, then replace k by $k+1$ and go to Step 11. Otherwise, replace Δ by 0.75Δ. If $\Delta < \varepsilon$, then stop; otherwise, go to Step 13.



$V_i^k = \{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4, \mathbf{v}_5\}$

Figure 2. Voronoi Polygons for Four Sensors and Vertex Set V_i^k

Choosing a Set of Initial Solutions

Since SELP is a nonconvex programming problem, it has many local optima, and the choice of initial solution plays an important role in final solution quality. One option is to use many randomly generated initial solutions. However, through empirical testing, it was determined that initial solutions generated by uniformly distributing sensors along the boundary of *S* typically generated high quality solutions. Thus, this was the tactic used to initiate TLP. Four types of starting solution sets were generated. Let *L* represent the longest side of *S* and let $\mathbf{x}_c = (x_c, y_c)$ represent the mean of the vertices of *S*. In a Type 1 initial solution, the *m* sensors are uniformly distributed along the boundary of *S* with the first sensor located on side *L*. Five different such starting solutions are generated by varying the position of the first sensor on *L*. The first sensor is located at five different positions uniformly positioned along side *L*. Finally the sensors are moved from the

boundary toward the center point $\mathbf{x}_c$. That is, if $\mathbf{x}_b$ represents a boundary point, the corresponding sensor location will be $0.75\mathbf{x}_b + 0.25\mathbf{x}_c$.

A Type 2 starting solution is generated in basically the same way as Type 1 except that one of the sensors is located at $\mathbf{x}_c$ with the remaining $m-1$ located along the boundary of *S* as in Type 1. In this case the boundary locations are repositioned using $0.80\mathbf{x}_b + 0.20\mathbf{x}_c$. Again, five such solutions are generated.

A Type 3 starting solution is generated in the same way as Type 2 except two sensors are located along the longest diagonal of *S* with the remaining $m-2$ distributed along the boundary. If $\mathbf{x}_d$ and $\mathbf{x}_e$ represent the endpoints of the longest diagonal, the two interior points are located at $0.60\mathbf{x}_d + 0.40\mathbf{x}_e$ and $0.40\mathbf{x}_d + 0.60\mathbf{x}_e$. Again, five solutions are generated, in this case the boundary solutions are repositioned using $0.85\mathbf{x}_b + 0.15\mathbf{x}_c$.

In a Type 4 solution, the sensors are divided into two groups when the number of sensors *m* is greater than $n+1$ where *n* is the number of vertices of *S*. Since *S* has *n* vertices and vertices are critical points in the computational of the maximum probability, a set of *n* sensors are uniformly located along the boundary as in a Type 1 solution and repositioned using $0.90\mathbf{x}_b + 0.10\mathbf{x}_c$. The remaining $m-n$ sensors are also located along the boundary as Type 1 and repositioned using $0.50\mathbf{x}_b + 0.50\mathbf{x}_c$. Three different positions are used for each set of sensors and are then combined for a total of nine solutions.

Thus, a total of (5 + 5 + 5 + 9) = 24 starting solutions were used.

Figure 3 provides an example of each type of starting solution when there are nine sensors $(m = 9)$ and *S* is a square $(m = 9)$.

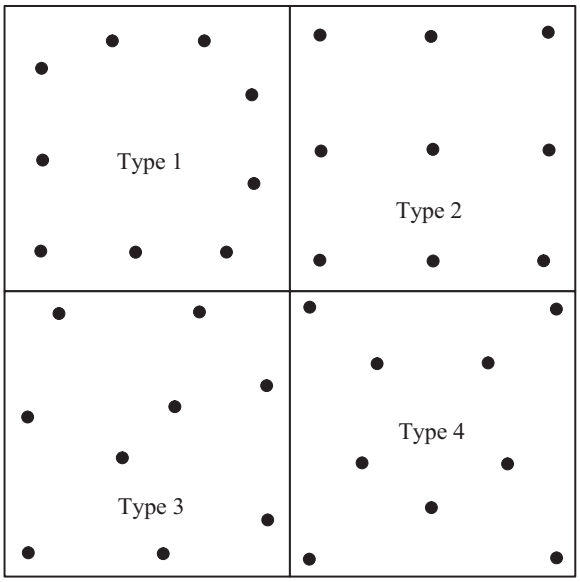


Figure 3. Examples of Starting Solutions

Figure 4 provides a graphical depiction of an initial Type 1 solution and the first five iterations of the TLP algorithm for a 5-sensor problem on a square.

Choosing a Step Length Δ

The choice of the step length Δ in Step 6 of Phase I and Step 13 of Phase II also has a significant impact on the efficiency of the algorithm as well as the quality of the final solution. If the initial Δ is small, then many additional iterations may be necessary and the local optimum found will likely be near the initial solution. Whereas if the initial Δ is large, it may be possible to find a local optimum that is some distance away from the initial solution, but there may be an excessive number of unsuccessful steps where the objective does not improve. Consider Phase 1. Let $\mathbf{x}_1^k, ..., \mathbf{x}_m^k$ represent the current solution and for each $\mathbf{x}_i^k$, let $V_{i\max}^k = \left\{\mathbf{v}_j \in V_i^k : q_j = q_{i\max}^k\right\}$ as in Step 4. Let $d_{\min}^k = \min_{i=1,...m}\left\{\min_{\mathbf{v}_j \in V_{i\max}^k}\left\{d(\mathbf{v}_j, \mathbf{x}_i^k\right\}\right\}$. That is $d_{\min}^k$ is the minimum distance between the sensor locations and their respective largest local peaks. Then, to allow the algorithm the opportunity to find multiple local optima based on a single initial solution, two initial step sizes were used, $d_{\min}^k$ and $9d_{\min}^k$. Note, however, that in Step 6, the step length may be reduced by multiples of 0.75 until a successful step is

taken. Let Δ^k be the actual successful step length used at iteration k. Then since the step lengths of successive iterations generally decrease during the course of the algorithm, two step lengths rules were used for iteration $k+1$: $\min\left\{6\Delta^k, d_{\min}^{k+1}\right\}$ and $\min\left\{6\Delta^k, 9d_{\min}^{k+1}\right\}$. These parameter settings were chosen using empirical testing. So, in essence, Phase I is solved twice, one with a small initial step length and once with a large initial step length. Both of these are followed by Phase II, in which the step length rule $\min\left\{6\Delta^k, d_{\min}^{k+1}\right\}$ was used.

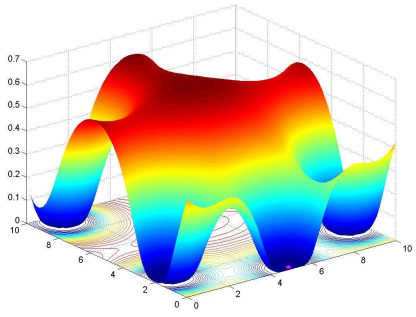
4. Computational Results: The TLP algorithm was coded in FORTRAN and compiled with Compaq Visual FORTRAN Professional Edition 6.6.B. For comparison, the routine Fminimax in the Matlab Optimization Toolbox was also used to solve SELP. Fminimax is specifically designed to solve minimax-type problems. A coarse 50×50 grid was used to discretize the set S and evaluate the maximum probability of non-detection. This grid size was chosen to attain a reasonable resolution (0.2 based on a 10×10 square) while also taking into account cpu time. Ten randomly generated starting solutions were used for each run of Fminimax, which was compiled using Matlab Compiler version 7.0.

Both programs were run under identical conditions on a PC with an Intel Pentium 4 2.4 Ghz CPU with 512 MB of RAM. For comparison, the probability of non-detection for the final solution of each algorithm was determined by utilizing a 1000×1000 grid. Table 1, provides a summary of the results when S is a 10×10 square and the dpf is gravity decay with $\pi(d) = 1-e^{-3/d^2}$. Considering all 29 problems, the average cpu time for TLP was 19.49 seconds whereas Fminimax required an average of 524.82 seconds. TLP attained a better final probability of non-detection in 17 of the 29 cases.

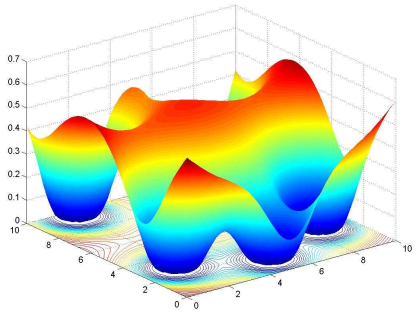
Both algorithms were also used to find sets of sensors to monitor a polygon when the dpf was gravity decay defined by $\pi(d) = 1-e^{-3/d^2}$. The polygon was a six-sided polygon with vertex set {(0,0), (6,1), (10,4), (10,6), (6,9), (1,4)}, and Tables 2 summarizes the results. Note that TLP yielded the better final solution in 17 of the 19 problems solved. TLP required an average of 11.62 seconds of cpu time whereas Fminimax used an average of 222.63 seconds. Additional computational testing can be found in Cavalier, *et al*. [2005].

5. Conclusion: This paper presented a heuristic algorithm for determining the location of a finite number of identical sensors to detect an event in a given planar region. The planar region was assumed to be a convex polygon and the objective is to minimize the maximum probability of non-detection. The problem is a difficult nonlinear nonconvex programming problem with a multitude of applications. The heuristic algorithm utilizes Voronoi polygons to estimate the probability of non-detection and to determine a search direction. Computational results demonstrated that the algorithm is relatively fast and generates high-quality solutions.

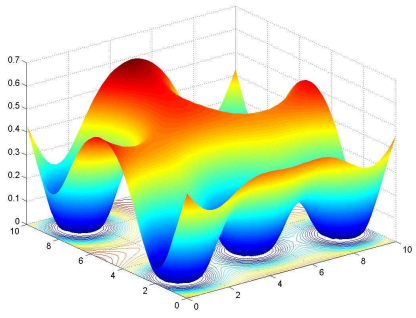
6. Acknowledgment: This material is based upon work supported by the National Science Foundation under Grant No. 0400140. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.



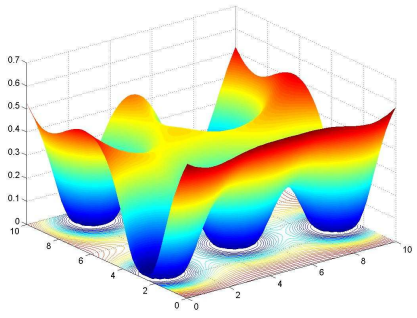
Initial Type 1 Solution



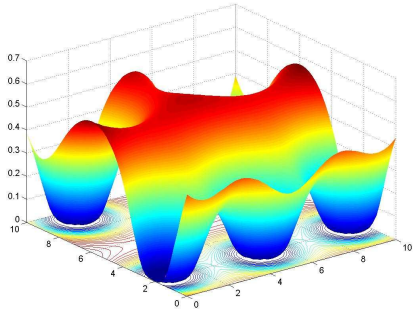
Iteration 1



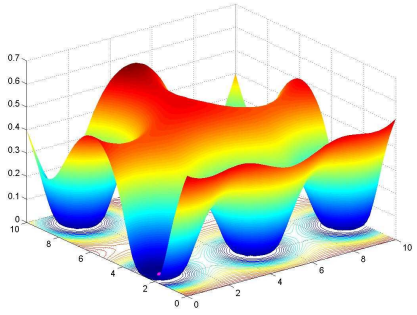
Iteration 2



Iteration 3



Iteration 4



Iteration 5

Figure 4: Graphical Depiction of TLP Algorithm

Table 1: Results for gravity decay $\pi(d) = 1 - e^{-3/d^2}$ on a 10×10 square

Number of Sensors	TLP Algorithm		MATLAB Fminimax		Difference $P_{TLP} - P_{Fminimax}$	(% Difference) $\frac{100*(P_{TLP} - P_{Fminimax})}{\min(P_{TLP}, P_{Fminimax})}$
	Probability of Non-detection P_{TLP}	CPU Time (secs)	Probability of Non-detection $P_{Fminimax}$	CPU Time (secs)		
2	0.865889	0.08	0.865890	29.48	–0.000001	0.00
3	0.797697	0.16	0.796920	32.66	0.000777	0.00
4	0.587497	0.29	0.587680	50.63	–0.000183	0.00
5	0.516551	0.48	0.513140	83.73	0.003411	–0.42
6	0.411270	1.16	0.409480	117.94	0.001790	–0.69
7	0.343288	1.81	0.341040	163.68	0.002248	–0.08
8	0.238105	2.30	0.238670	167.88	–0.000565	–0.03
9	0.180115	3.41	0.179560	225.04	0.000555	–1.26
10	0.148831	4.10	0.143860	207.65	0.004971	–2.23
11	0.108221	5.47	0.108110	263.77	0.000111	–0.89
12	0.082140	4.91	0.082084	300.63	0.000056	–0.87
13	0.055696	7.84	0.056095	411.23	–0.000399	–0.44
14	0.042850	8.46	0.041834	480.5	0.001016	–1.86
15	0.032697	12.94	0.032117	426.99	0.000580	–2.96
16	0.026663	10.10	0.023778	502.77	0.002885	–3.81
17	0.019994	14.66	0.017808	541.54	0.002186	–3.70
18	0.012355	14.87	0.012470	589.28	–0.000115	–2.08
19	0.009224	20.64	0.009837	655.79	–0.000613	–4.27
20	0.005885	17.13	0.006901	697.59	–0.001016	–3.66
21	0.004487	29.68	0.005141	773.89	–0.000654	–4.28
22	0.002952	29.14	0.003564	857.53	–0.000612	–1.49
23	0.002262	31.68	0.002886	812.08	–0.000624	–3.94
24	0.001773	30.69	0.002023	959.1	–0.000250	–6.26
25	0.001124	43.44	0.001468	983.97	–0.000345	–4.29
26	0.000754	43.65	0.001512	970.27	–0.000758	–3.75
27	0.000604	54.45	0.001170	1091.89	–0.000566	–8.65
28	0.000474	50.96	0.001047	907.42	–0.000573	–22.25
29	0.000232	62.34	0.000801	933.48	–0.000569	–5.12
30	0.000161	58.43	0.000681	981.38	–0.000520	–4.60

Table 2: Results for gravity decay $\pi(d) = 1 - e^{-3/d^2}$ on a six-sided polygon

Number of Sensors	TLP Algorithm		MATLAB Fminimax		Difference $P_{TLP} - P_{Fminimax}$	(% Difference) $\frac{100*(P_{TLP} - P_{Fminimax})}{\min(P_{TLP}, P_{Fminimax})}$
	Probability of Non-detection P_{TLP}	CPU Time (secs)	Probability of Non-detection $P_{Fminimax}$	CPU Time (secs)		
2	0.723580	0.16	0.728530	18.08	–0.004950	–0.68
3	0.583972	0.30	0.586960	33.18	–0.002988	–0.51
4	0.426887	0.52	0.426470	53.37	0.000417	0.10
5	0.267728	0.99	0.275630	53.17	–0.007902	–2.95
6	0.190962	1.22	0.190430	106.74	0.000532	0.28
7	0.118905	1.72	0.129160	131.25	–0.010255	–8.62
8	0.068871	3.75	0.074820	139.63	–0.005949	–8.64
9	0.044392	5.15	0.048702	162.17	–0.004310	–9.71
10	0.026441	6.96	0.029896	229.45	–0.003455	–13.07
11	0.016524	8.69	0.018382	210.39	–0.001858	–11.24
12	0.009666	9.95	0.010921	305.81	–0.001255	–12.98
13	0.005254	12.68	0.006854	313.12	–0.001600	–30.46
14	0.002665	13.22	0.004049	330.32	–0.001384	–51.95
15	0.001550	17.20	0.002734	349.84	–0.001184	–76.43
16	0.000878	19.90	0.001995	440.49	–0.001117	–127.11
17	0.000519	22.99	0.001331	309.09	–0.000812	–156.30
18	0.000252	26.78	0.000831	350.18	–0.000579	–229.68
19	0.000151	31.83	0.000653	306.72	–0.000502	–331.31
20	0.000087	36.84	0.000522	387.04	–0.000435	–500.28

7. References:

Aurenhammer, F. [1991] Voronoi Diagrams – a Survey of a Fundamental Geometric Data Structure, *ACM Computing Surveys* 23:345-405.

Brady, S.D., R.E. Rosenthal, D. Young [1983] Interactive Graphical Minimax Location of Multiple Facilities with General Constraints, IIE Transactions 15:242-253.

Cavalier, T.M., W.A. Conner, E. del Castillo, S.I. Brown [2005] A Heuristic Algorithm for Minimax Sensor Location in the Plane, *European Journal of Operational Research*, under review.

Charalambous, C. [1981] Iterative Algorithm for the Multifacility Minimax Location Problem with Euclidean Distances, *Naval Research Logistics Quarterly*, 28(2):325-337.

Chatelon, J.A., D.W. Hearn, T.J. Lowe [1982] A Subgradient Algorithm for Certain Minimax and Minisum Problems – the Constrained Case, *SIAM Journal on Control and Optimization* 20:455-469.

Demjanov, V.F. [1968] Algorithms for Some Minimax Problems, *Journal of Computers and System Science* 2:342-380.

Drezner, Z. [1982] On Minimax Optimization Problems, *Mathematical Programming* 22:227-230.

Drezner, Z., G.O. Wesolowsky [1978] A New Method for the Multifacility Minimax Location Problem, *Journal of the Operational Research Society* 29:1095-1101.

Drezner, Z., G.O. Wesolowsky [1997] On the Best Location of Signal Detectors, *IIE Transactions* 29(11):1007-1015.

Elzinga, D.J., D.W. Hearn [1972] Geometrical Solutions for Some Minimax Location Problems, *Transportation Science* 6:379-394.

Elzinga, D.J., D.W. Hearn, W.D. Randolph [1976] Minimax Multifacility Location with Euclidean Distances, *Transportation Science* 10:321-336.

Howitt, I., S.-Y Ham [1999] Base Station Location Optimization, *IEEE Vehicular Technology Conference* 4:2067-2071.

Iri, M., K. Murota, T. Ohya [1984] A Fast Voronoi-Diagram Algorithm with Applications to Geographical Optimization Problems, *Lecture Notes in Control and Information Sciences* 59:273-288.

Love, R.F., G.O. Wesolowsky, S.A. Kraemer [1973] A Multifacility Minimax Location Method for Euclidean Distances, *International Journal of Production Research* 11:32-40.

Love, R.F., J.G. Morris, G.O. Wesolowsky [1988] *Facilities Location: Models and Methods*, North Holland, NY.

Okabe, A.; Boots, B.; and Sugihara, K. [1992] *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, New York: John Wiley and Sons Ltd.

Perez, R. [1998] Propagation Effects on Interference in Wireless Communication Services, *IEEE International Symposium on Electromagnetic Compatibility*, 1:86-91.

Plastria, F. [1995] Continuous Location Problems, in *Facility Location: A Survey of Applications and Methods*, Edited by Z. Drezner, Springer-Verlag, New York, pp. 225-262.

Polak, E., D.Q. Mayne, J.E. Higgins [1991] Superlinearly Convergent Algorithms for Min-Max Problems, *Journal of Optimization Theory and Algorithms* 69(3):407-439.

Preparata, F.P. and Shamos, M.I. [1985] *Computational Geometry: An Introduction*, New York: Springer-Verlag.

Saunders, Simon R. [1999] Antennas and propagation for wireless communication systems, New York: J. Wiley.

Shamos, M.I., D. Hoey [1975] Closest Point Problems, *Proceedings of the 16th Annual Symposium on Foundations of Computer Science*, 151-162.

Suzuki, A., A. Okabe [1995] Using Voronoi Diagrams, in *Facility Location: A Survey of Applications and Methods*, Edited by Z. Drezner, Springer-Verlag, New York, pp. 103-118.

Suzuki, A., Z. Drezner [1996] The *p*-Center Location Problem in an Area, *Location Science* 4(1/2):69-82.

Tutschku, K. [1998] Demand-Based Radio Network Planning of Cellular Mobile Communication Systems, *Proceedings - IEEE INFOCOM* 3:1054-1061.

Vardi, A. [1992] New Minimax Algorithm, *Journal of Optimization Theory and Applications* 75(3):613-634.