9-1-2004

# A Distributed Evolutionary Algorithmic Approach to the Coverage Problem for Submersible Sensors

Jason C. Tillett
*Rochester Institute of Technology*

Raghuveer Rao
*Rochester Institute of Technology*

Ferat Sahin
*Rochester Institute of Technology*



## Recommended Citation

Jason C. Tillett, Raghuveer Rao, Ferat Sahin, "A distributed evolutionary algorithmic approach to the coverage problem for submersible sensors", Proc. SPIE 5417, Unattended/Unmanned Ground, Ocean, and Air Sensor Technologies and Applications VI, (1 September 2004); doi: 10.1117/12.541664; https://doi.org/10.1117/12.541664

# A distributed evolutionary algorithmic approach to the coverage problem for submersible sensors

Jason Tillett , Raghuveer Rao and Ferat Sahin
Rochester Institute of Technology, Rochester NY 14623

## ABSTRACT

Untethered, underwater sensors, deployed for event detection and tracking and operating in an autonomous mode will be required to self-assemble into a configuration, which optimizes their coverage, effectively minimizing the probability that an event in the target area goes undetected. This organized, cooperative, and autonomous, spreading-out of the sensors is complicated due to sensors localized communication. A given sensor will not in general have position and velocity information for all sensors, but only for those in its communication area. A possible approach to this problem, motivated by an evolutionary optimization technique, Particle Swarm Optimization (PSO) is proposed and extended in a novel way. A distributed version of PSO is developed. A distributed version of PSO is explored using experimental fitness to address the coverage problem in a two dimensional area.

## 1. INTRODUCTION AND RELATED WORK

Blanket coverage [1], refers to the ability of the collection of mobile nodes to provide surveillance or measurements over a specified area or region. The key problem for this type of coverage is to ensure that all required areas are adequately covered. This involves provision of appropriate node density while attending to considerations of economy, that is, avoid overpopulating these areas. Coverage problems are beginning to draw some attention and some current approaches use Voronoi tessellations of the nodes to optimize a metric, like finding a maximal breach path. These algorithms are only applicable to static deployments [2, 3]. The blanket coverage problem can be categorized under the topic of locational optimization problems and a review of Voronoi diagrams applied to such problems can be found in [4]. Blanket coverage has been implemented for mobile nodes/robots using virtual forces due to other nodes and obstacles [5], but it appears that their implemented approach requires nodes to be initialized more densely than their target density and there is no mechanism for drawing nodes together, only repulsion is supported. The possible target application for their approach is search and rescue in an indoor hazardous environment. A fluid environment introduces complications to the coverage problem. In a fluid environment, advection may require nodes to come together to maintain the optimal coverage. An attractive tendency must therefore be present as well. Another force based-approach to coverage optimization is very similar to this work [6] in that they use node density imbalances to generate forces. Their work confines the nodes while we assume that the nodes are deployed in an unbounded plane. Additionally, they have fixed the sensing and communication ranges while we explore their impact. In another work, coordinated algorithms for vehicle control capable of solving the blanket coverage problem have been presented but each node must exactly calculate its Voronoi cell. [7] A distributed approach [8] to calculating the Voronoi region is developed. Knowing the Voronoi region of each node empowers the distributed algorithm designer to accurately maneuver the nodes and in this paper we explore using approximations of the Voronoi regions. Formations are another form of coverage. Efforts to autonomously and cooperatively urge collections of mobile robots into various configurations like circles, lines and polygons are ongoing. [9] With respect to submersible mobile sensing platforms, maintaining formations can optimize data collection capabilities of the collection. [10]

The main goals of this paper are to introduce a new distributed optimization paradigm, formulate and apply the approach to the blanket coverage problem in the context of submersible sensors and characterize the results. The problem is stated in section 2. In section 3 the Particle Swarm Optimization (PSO) algorithm is discussed. A distributed formulation of the algorithm is introduced and developed in section 4. In section 4.1 the Distributed Particle Swarm Optimization (DPSO) approach is cast for the coverage problem. The simulations and results are discussed in section 5 followed by a summary and future work.



Unattended/Unmanned Ground, Ocean, and Air Sensor Technologies
and Applications VI, edited by Edward M. Carapezza, Proc. of SPIE Vol. 5417
(SPIE, Bellingham, WA, 2004) · 0277-786X/04/$15 · doi: 10.1117/12.541664

## 2. PROBLEM STATEMENT

We seek to minimize area within a target area that remains uncovered. If an event occurs in an uncovered area, the event is undetected. Equivalently, we seek to minimize the occurrence of undetected events. If the $i^{th}$ submersible sensor can cover an area $A_i$, then the total area covered by the sensors is $A = \bigcup_{i=1}^{N} A_i$. If the target area is given by $T$, then we seek the solution to $\min\left(T - (T \cap A)\right)$.

## 3. THE APPROACH: PARTICLE SWARM OPTIMIZATION (PSO)

The PSO [11] approach utilizes a cooperative swarm of particles, where each particle represents a candidate solution, to explore the space of possible solutions to the optimization problem of interest. Each particle is randomly or heuristically initialized and then allowed to 'fly'. At each step of the optimization, each particle is allowed to evaluate its own fitness and the fitness of its neighboring particles. The fitness or objective function is a function of the solution. Each particle can keep track of its own solution, which resulted in the best fitness, as well as see the candidate solution for the best performing particle in its neighborhood. At each optimization step, each particle adjusts its candidate solution (flies) according to,

$$\begin{aligned} v(t+1) &= v(t) + \phi_1(x - x_p) + \phi_2(x - x_n) \\ x(t+1) &= x(t) + v(t+1) \end{aligned} \tag{1}$$

Subscripts for particle index and dimensionality have been left off of Eqn. 1, which may be interpreted as the 'kinematic' equation of motion for one of the particles (test solution) of the swarm where the particle is one-dimensional. The variables in Eqn. 1 are summarized in Table 1.

Table 1- List of variables used in the equations

| | |
|---|---|
| $v$ | The particle velocity. |
| $x$ | The particle position (test solution). |
| $t$ | Time |
| $\phi_1$ | A uniform random variable usually distributed over [0,2]. |
| $\phi_2$ | A uniform random variable usually distributed over [0,2]. |
| $x_p$ | The particle's position (previous) that resulted in the best fitness so far. |
| $x_n$ | The neighborhood position that resulted in the best fitness so far. |

Eqn. 1 can be interpreted as follows. Particles combine information from their previous best position and their neighborhood best position to maximize the probability that they are moving toward a region of space that will result in a better fitness.

There is a problem with using PSO directly to solve the coverage problem we have described. Each node must move autonomously. There is no communication with a base station that can perform the optimization and pass along optimal communication ranges to each of the nodes. Thus, even if we could construct a PSO particle representing the solution and construct an appropriate fitness, the nodes could not compute the result. We need a distributed version of the PSO algorithm. [12]

## 4. DISTRIBUTED PSO (DPSO)

In traditional PSO, the fitness function is shared among the particles in the swarm. Particles in traditional PSO represent the solution to a single optimization problem. In contrast, in the distributed form developed here, particles have no knowledge, or limited knowledge, of the global objective function. Particles do not represent a global solution to a single optimization problem. Rather, particles have individual objectives and their objective function is a function of their individual parameters. This can be written as,

$$f_i\left(p_{i1}, p_{i2}, \ldots p_{iM},\right). \tag{2}$$

where each particle, $i$, has $M$ parameters. The operational parameters, $p_{ij}$, of Eqn. 2 can be: communication range, sensing range, carrier sense range, number of neighbors, battery reserve level. This list provided is exemplary and is neither complete nor the list used in this paper. The system designer may and probably will have a global objective or optimization targeted, but the particles cannot evaluate the global objective function because they do not have access to all of the particles' parameters. It is up to the designer to craft a suitable local objective function that will cause the system to approximate the desired global objective.

Another difficulty arises due to the use of local objective functions and their dependence on local parameters. How does one calculate the neighborhood best (labeled $x_n$ in Eqn. 1)? In traditional/centralized PSO, the neighborhood best is simply the solution represented by the most fit neighbor. In DPSO, the solution represented by the most fit neighbor evaluated using the particle's local objective function, may not, and probably won't, result in a better fitness for the particle. Therefore particles must be able to interpret solutions/parameters received from their neighbors. Particles must be able to convert parameter values and fitnesses exchanged with neighbors into a possible "better fit" set of values for their own operational parameters, their neighborhood best parameters. This mapping is expressed as $\tilde{Q}_i \Rightarrow \vec{p}_{i,nbest}$ where

$$\tilde{Q}_i = \left\{ \begin{array}{l} (f_1, p_{11}, p_{12} \ldots p_{1M}), \\ (f_2, p_{21}, p_{22} \ldots p_{2M}), \\ \ldots \\ (f_{N_i}, p_{N_i 1}, p_{N_i 2} \ldots p_{N_i M}) \end{array} \right\} \tag{3}$$

and $\vec{p}_{i,nbest}$ is the neighborhood best values for node $i$. $\tilde{Q}_i$ is a set, for node $i$, of values of fitnesses, $f$, and parameters, $p$, that it receives from its $N_i$ neighbors. In Eqn. 3, each neighbor has a single fitness value, but may have up to $M$ parameter values to report to particle $i$. Once each particle is able to evaluate its fitness function and is able to construct its $\tilde{Q}$ set with information from its neighbors, the computation can proceed as in traditional PSO. Eqn. 1 becomes

$$\begin{array}{l} \vec{v}_i(t+1) = \vec{v}_i(t) + \phi_1(\vec{p}_i - \vec{p}_{i,best}) + \phi_2(\vec{p}_i - \vec{p}_{i,nbest}) \\ \vec{p}_i(t+1) = \vec{p}_i(t) + \vec{v}_i(t+1) \end{array} . \tag{4}$$

We have used slightly modified notation in Eqn. 4 in order to remove some labeling ambiguity. The subscript *best* denotes the previous best value for the particle, which determines the cognitive component of the particles' motions. The subscript *nbest* denotes the neighborhood best and determines the social component of the particles' motions. We changed $x$'s in Eqn. 1 to $p$'s to be consistent with the notation in Eqns. 2 and 3. We emphasize here that the calculation of the neighborhood best must be discussed in the context of a specific problem.

### 4.1 DPSO for coverage optimization

The Voronoi representation of a snapshot of the submersible mobile sensor network with sensor nodes as the generators of each Voronoi region would create regions around nodes where no other nodes are found. Everywhere within a given node's Voronoi region is closer to the generating node than to all other nodes. If we approximate a given node's Voronoi region by a circle or radius $r_i$, then the local node density will be given by $\left(\pi r_i^2\right)^{-1}$. We can articulate what we mean by a circle approximating a Voronoi region as follows. We seek a circle about the generating node such that the sum of the area that is inside the circle but outside the Voronoi region and the area that is inside the Voronoi region but outside the circle is minimized. For regular polygons, an analytical expression for the radius of the circle may be derived. For arbitrary polygons, a numerical approach must be used to find the circle. The numerical search for the optimum approximating circle is simplified by the fact that the polygons are convex (Voronoi regions). This is because if $g(r)$ is the sum length of the portions of the circle that are inside the polygon, then $g(r)$ is monotonic and decreasing. This makes the extremum function single valued and a hill climbing algorithm will converge optimally.

A first order approximation of a Voronoi tessellation could consist of the set of approximating circles. For nodes distributed in a regular hexagonal grid, the resulting set of approximating circles will have the property that each circle will be tangent to neighboring circles. Again, as a first order approximation, for uniformly distributed nodes, we can approximate the Voronoi tessellation by searching for a set of circles that has the property that each circle is tangent to every neighboring circle. Since a set is likely not to exist, we seek to minimize the deviation from this desired property. This objective can be internalized in each node through the fitness function,

$$f_i = \sum_{j \neq i} (d_{ij} - r_i - r_j)^2 \,, \tag{5}$$

where the sum is over neighbor nodes of $i$ that are within communication range, $d_{ij}$ is the distance separating the nodes. The fitness function in Eqn. 5 will attempt to fit the nodes with tangent circles centered on the nodes.

The $\tilde{Q}_i \Rightarrow \vec{p}_{i,nbest}$ mapping is simple. Neighboring nodes exchange fitness values and a parameter set. Here only one parameter need be exchanged and that is $r_i$. Each node will assume that the fittest neighbor has the best approximation of its radius and so will adopt as the neighborhood defined best value for it own $r_i$ to be

$$r_{i,nbest} = abs\left|d_{ij^*} - r_{j^*}\right| . \tag{6}$$

The asterisk denotes the neighbor of node $i$ that is the most fit. The important revelation is that each node is trying to optimize its own value of $r_i$, but is taking cues about what its value of $r_i$ should be based on the fitnesses and $r_j$'s reported by its neighbors. Figure 1 illustrates a set of 25 fixed nodes estimating the radius of their Voronoi regions. That would be the end of the formulation if the nodes were fixed. However, our nodes are mobile and are tasked to spread out to optimize coverage of a target area. We need to discuss next how an approximation of the Voronoi tessellation can be used to control the motion of the nodes.

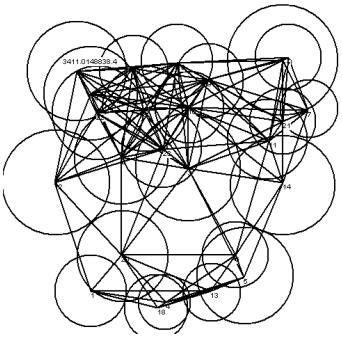


Figure 1 – Static nodes estimate the size of their Voronoi regions.

We adopt a simple model to control the motion of the nodes. We assume that neighboring nodes are connected by virtual, modified springs whose equilibrium position is determined by a design parameter, $R_S$, which is the adjusted sensing range of the nodes. The adjusted sensing range will typically be fixed by the sensing hardware. The adjusted sensing range is related to the actual sensing range. For a collection of nodes to obtain 100% coverage per unit area when hexagonally packed such that their adjusted sensing range circles are mutually tangent, their actual sensing range must be larger than their adjusted sensing range in order to cover gaps between the circles. The actual sensing range must be $2/\sqrt{3}$ times the adjusted sensing range. We also assume for our simulations that the relative positions of the nodes are known so that neighboring nodes may calculate component forces due to the virtual, modified springs along arbitrary axes. This assumption is not necessarily a condition limiting the practical applicability because it is possible to integrate directional sensors into the hardware. We adopt for the force on node $i$ due to neighboring node $j$

$$F_i = \begin{cases} \Delta e^{-\frac{\Delta}{R_S}} & \Delta \geq 0 \\ \frac{3\Delta}{2} e^{-\frac{\Delta}{R_S}} & \Delta < 0 \\ 0 & |F_i| < 1 \end{cases} \tag{7}$$

where

$$\Delta = D_{ij} - r_i' - r_j' \tag{8}$$

and the primes in Eqn. 8 denote that the values are averaged over the last 10 steps of the algorithm. This is done to make the algorithm more amenable to application to a physical system where control limitations bound the response of the node. A positive force is attraction and a negative force is repulsion. Note that the node velocity is bounded to mimic limitations found in a physical system. Now the nodes can experience forces to enable them to react and attempt to optimize Eqn. 5, but without an additional piece to the node's fitness, it will not be motivated to spread out as desired.

The missing piece of the fitness includes a component to encourage the nodes to be regularly spaced. When nodes are regularly spaced, their values for $r_i$ are equal. The final component to the nodes fitness will be to encourage the node to adopt a value for $r_i$ that is equal to its adjusted sensing radius, $R_S$. So finally, the fitness function of each node is taken to be

$$f_i = (R_S - r_i)^2 + \frac{1}{N_i} \sum_{j \neq i} (d_{ij} - r_i - r_j)^2 + (r_i - r_j)^2, \tag{9}$$

where $N_i$ is the number of neighbors for node $i$.

## 5. SIMULATIONS AND RESULTS

There is an extensive list of design choices in our simulations. It is not possible for us to report on the exhaustive simulation space here so we will itemize some choices we have fixed. The force function is fixed. The form of this function can dramatically affect the simulation results. We have chosen our form through experimentation. The simulation environment is an infinite plane. The adjusted sensing range of the nodes is set at 15 and nodes are uniformly and randomly distributed about the origin on the interval [-50,50]. We choose to experiment with 25 nodes initially.

We will vary some aspects of the simulation. The radius of the approximate Voronoi regions, $r_i$, are to be found using DPSO. We offer as an alternative a simple heuristic approach to determine a radius. We average the midway point between the node's nearest and farthest neighbor. We call this the heuristic approach to determining the radius. It is offered in contrast to the DPSO method. Both methods of determining a radius rely on a topology, which is in turn determined by the communication radius, $R_C$ (not the same as the approximate Voronoi radius). The way in which $R_C$ of the nodes is determined must also be specified. We offer two alternatives to determining the topology. In the first method, the $R_C$'s of all of the nodes are fixed and identical throughout the simulation. The second approach allows the nodes to adapt individually their value for $R_C$ during the simulation. The second method uses our DPSO power control algorithm to achieve a topology. [13]

Our first batch of results presented here utilizes a fixed $R_C$ for each node. If $R_C < R_S$, then nodes will be unable to optimally spread out. Therefore, we adopt $R_C = R_S$ as the minimum value for $R_C$. A simulation is executed, each time randomly resetting the nodes initial positions, for incrementally larger values for $R_C$. Two sets of simulations are executed. In one set, the heuristic approach to determining $r_i$ is used. In the second set, the DPSO method to determining $r_i$ is used. Snapshots of the simulation visualization are saved after 700 steps of the algorithm, which is enough time to allow the nodes to settle down into a relatively steady state. The snapshots for the 2 sets of simulation trials are shown in Table 2 and Table 3. We see the effect of varying the communication radius. For small $R_C \sim R_S$, since communication links must exist to generate forces, nodes are unable to adjust their positions adequately in the plane.

Table 2 – Heuristic (rule based) basis for force determination ($r_i$), fixed communication range, $R_C$ varied from $R_S$ up to $5R_S$ set of multipliers= (1,1.25,1.5,1.75,2.0, 2.25, 2.5, 2.75, 3.0, 3.25, 3.5, 3.75, 4.0, 5.0). All figures are snapshots taken after 700 steps of the coverage algorithm.

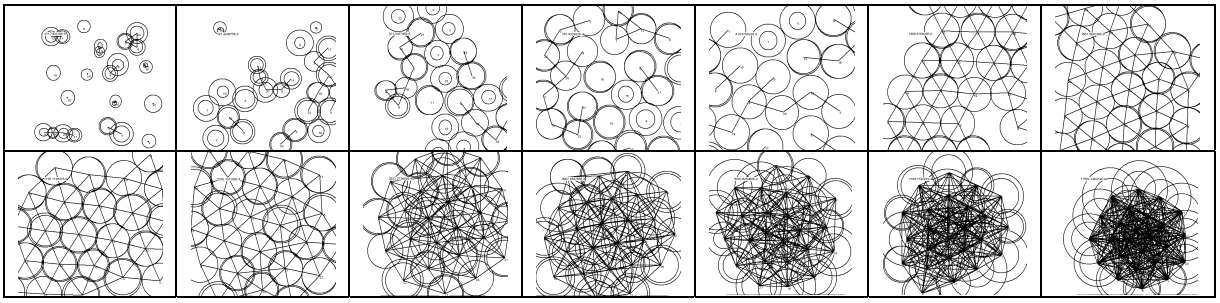
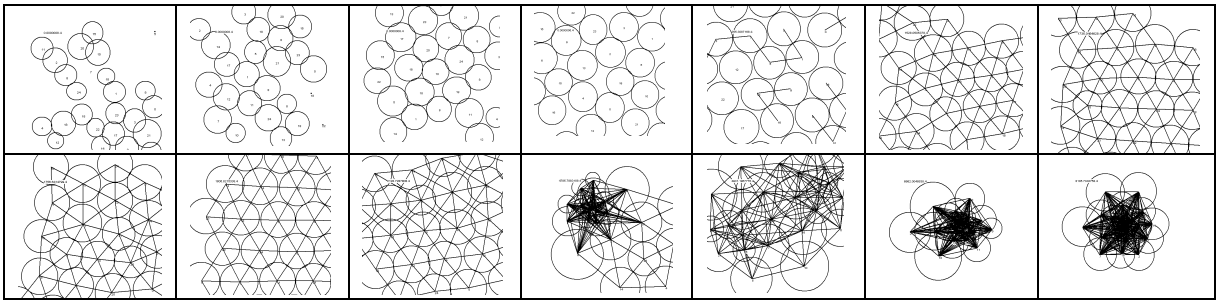


Table 3 -- DPSO based force determination ( $r_i$ ), fixed communication range, $R_C$ varied from $R_S$ up to $5R_S$, set of multipliers=(1,1.25,1.5,1.75,2.0, 2.25, 2.5, 2.75, 3.0, 3.25, 3.5, 3.75, 4.0, 5.0). All figures are snapshots taken after 700 steps of the coverage algorithm.



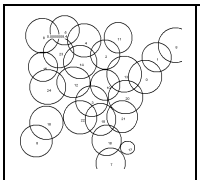
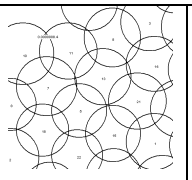
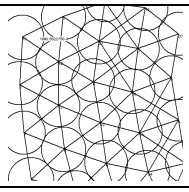
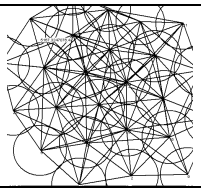
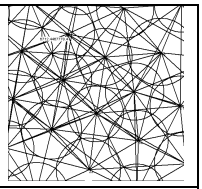
As $R_C$ is increased, nodes are able to optimize the coverage in the plane. It seems that it makes no difference whether $r_i$ is determined through DPSO or heuristically until $R_C = 3.25R_S$. The DPSO algorithm appears to hold its hexagonal packing over a larger range of $R_C$**.** Both simulation types show a collapse of the nodes for large $R_C$**.** The collapse of the nodes for large $R_C$ can be fixed for the DPSO approach with a simple change.

If we modify the expression for the neighborhood best *r* in the DPSO approach such that

$$r_{i,nbest} = r_{j^*}, \tag{10}$$

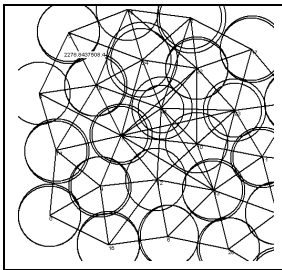
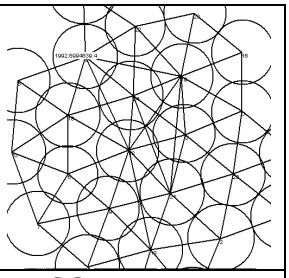
the behavior of the collection of nodes changes favorably. Table 4 shows the results of simulations executed with the simple modification to the neighborhood best.

Table 4 – DPSO *r* determination, simple neighborhood best vs. varying $R_C$

|  |  |  |  |  |
|---|---|---|---|---|
| $R_C = 1R_S$ | $R_C = 2R_S$ | $R_C = 3R_S$ | $R_C = 4R_S$ | $R_C = 5R_S$ |

An alternate approach to controlling the collapse of the nodes for large communication radius is to adaptively adjust the communication radii of the nodes. Although the DPSO approach to estimating $r_i$ benefits from the change in Eqn. 10, the heuristic approach remains the same. Both approaches could benefit from an adaptive power control algorithm. We have developed such an algorithm [13] which is also based on a DPSO approach but omit the details here. The coverage algorithm is now executed with power control enabled and the results for both the heuristic and DPSO approach to estimating $r_i$ are displayed side by side in Table 5. From the figures in Table 5, there appears to be little difference

Table 5 – Power control determined topology, snapshots after 700 steps

| | |
|---|---|
|  |  |
| heuristic $r$ | DPSO $r$ |

in the results obtained. It seems as long as power control is implemented, only a very basic estimation on the nodes approximate Voronoi region size is required. To further investigate whether the DPSO approach to estimating the approximate Voronoi region size offers any benefit, we ran 10 simulations of both approaches. We collected at each step of each simulation, the average separation of the nodes, the standard deviation of the separation between the nodes and the percent coverage of the collection of nodes. The average separation was calculated using only linked nodes. In other words, if the communication range of 2 nodes was such that they could communicate with each other, then their separation was included in the calculation of the average node separation. To calculate the percent coverage of the collection of nodes, we first define the target area. We define the target area as a square region. The maximum coverage of the nodes occurs when they are hexagonally packed. The area of 25 hexagons, whose equilateral triangular components measure $2R_S/\sqrt{3}$ on a side, is about the same as a square whose sides measure 197. We take our target area to be a conservative square, 150x150. All nodes are initialized on [-50,50] which is centered on the target square on the interval [-75,75]. The percent coverage can now be calculated using a Monte Carlo approach. At each step of the simulation, 100 events are randomly generated with a uniform distribution on [-75,75]. As each event is generated, the node collection is examined to determine if any node in the collection can detect the event. Detection occurs when the event is within the actual sensing radius of a node. If any node can detect the event, it is counted. The total number of detected events at each step is the percent coverage for that instant of the simulation. The statistics of the trials are presented in Table 6. The average node density begins at zero because we have initialized the nodes with a minimum communication range. When they communicate using the minimum range, it is very likely that no links will exist. Since the average node density is calculated only using separations for linked nodes, it is likely to begin at zero. Inspecting the figures in Table 6 we find that there are performance differences. For one, the DPSO approach achieves better uniform spacing of the nodes. This is evidenced by the convergence of the DPSO approach to smaller standard deviation in the node separation. Because the nodes achieve better spacing results, we expect better coverage. The results show that the DPSO approach converges to consistently higher values of the percent coverage. Not only does the DPSO approach achieve better coverage, but it achieves better coverage sooner. We see that after about 300 steps, the DPSO algorithm has achieved its maximum coverage while the heuristic approach is still working to improve the coverage.

## 6. SUMMARY AND FUTURE WORK

We have developed a scalable and robust approach to encouraging autonomous nodes to self-organize and spread-out. The algorithm uses a distributed extension of PSO to allow nodes to evolve such that they optimize their own local objective function. In doing so, a globally desirable behavior, such as blanket coverage emerges. Assuming that the devices have power control capabilities, the algorithm is resistive to problems associated with dense topologies. The force based approach to mobilization for coverage optimization hinges on an estimation of the local node density. This local node density can be identified as the inverse of the area of the node's Voronoi region. We use a distributed optimization technique to allow each node to estimate the size of its Voronoi region. Since we approximate each node's Voronoi region with a circle, much improvement could be garnered through the use of a more accurate approximating shape. An ellipse would easily model the actual Voronoi tessellation with a much-improved accuracy.

We implicitly assumed that the circles found by our distributed algorithm offered some approximation of the optimal circular approximation of a Voronoi region where the sum of the area that is inside the circle but outside the Voronoi region and the area that is inside the Voronoi region but outside the circle is minimized. There is no guarantee that the

circles found by our algorithm will converge to these optimal circles. We leave as further study, to fit an actual Voronoi tessellation with optimal circular approximations in order to derive properties for the sets of circles. These properties may then be incorporated into the fitness of Eqn. 9. Formations may also be stimulated using this approach through manipulation of the node's fitness function.

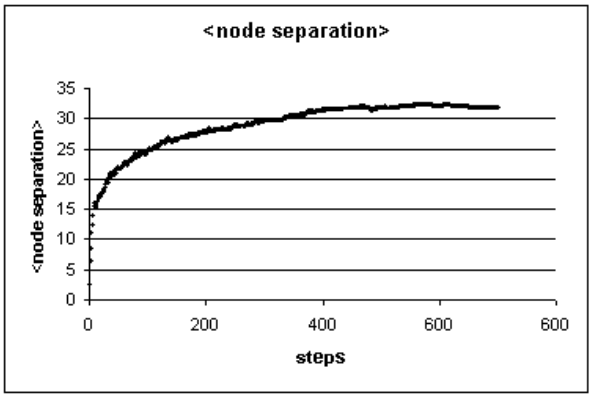
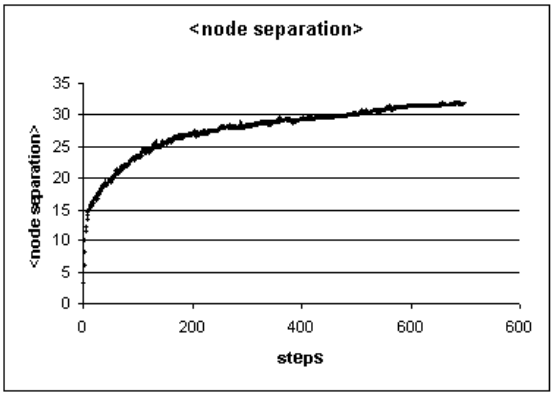
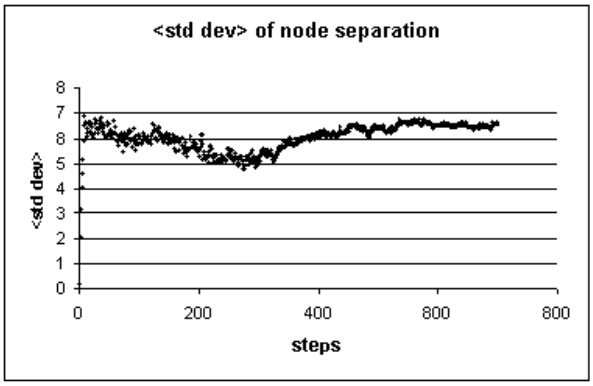
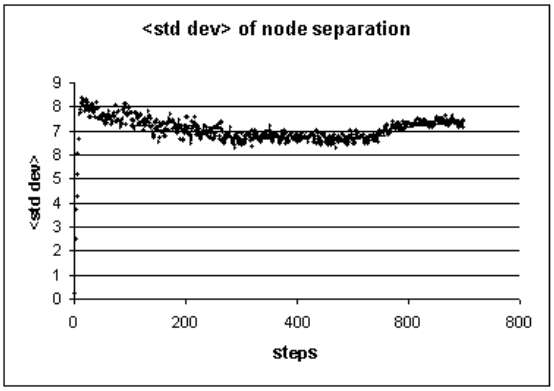
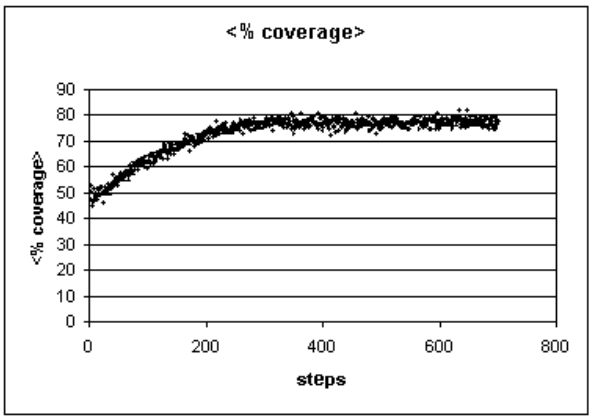
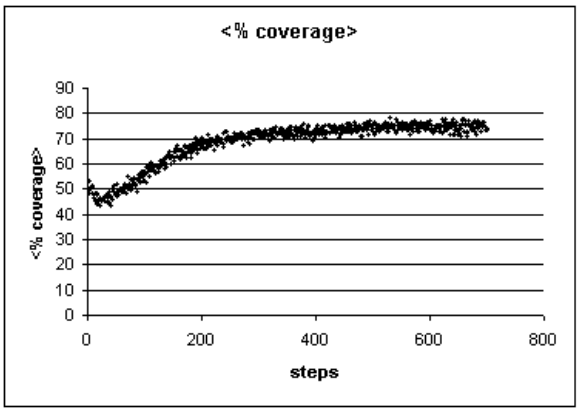
## ACKNOWLEDGEMENT

This research was conducted with the support of the Gleason Foundation through the Laboratory of Autonomous Cooperative Microsystems (LACOMS) at the Rochester Institute of Technology.

## REFERENCES

[1] D. W. Gage, "Command Control for Many-Robot Systems," presented at AUVS-92, the Nineteenth Annual AUVS Technical Symposium, Huntsville AL, 1992.
[2] X.-Y. Li, P.-j. Wan, and O. Frieder, "Coverage in Wireless Ad Hoc Sensor Networks," *IEE Transactions on Computers*, vol. 52, 2003.
[3] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava, "Coverage Problems in Wireless Ad-hoc Sensor Networks," presented at INFOCOM, 2001.
[4] A. Okabe and A. Suzuki, "Locational optimzation problems solved through Voronoi diagrams: a review," *European Journal of Operational Research*, vol. 98, pp. 445-56, 1997.
[5] A. Howard, M. J. Mataric, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: a distributed, scalable solution to the area coverage problem," presented at 6th International Symposium on Distributed Autonomous Robotics Systems (DARS02), Fukuoka, Japan, 2002.
[6] N. Heo and P. K. Varshney, "A distributed self-spreading algorithm for mobile wireless sensor networks," presented at IEEE Systems Man and Cybernetics, Washington, DC, 2003.
[7] J. Cortes, S. Martnez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," presented at IEEE International Conference on Robotics and Automation, Arlington, VA, 2002.
[8] M. Cao and C. Hadjicostis, "Distributed algorithms for Voronoi diagrams and applications in ad hoc networks," preprint 2002.
[9] K. Sugihara and I. Suzuki, "Distributed algorithms for formation of geometric patterns with many robots," *Journal of Robotic Systems*, vol. 13, pp. 127-39, 1996.
[10] E. Fiorelli, P. Bhatta, and N. E. Leonard, "Adaptive sampling using feedback control of an autonomous underwater glider fleet," presented at 13th International Symposium on Unmanned Submersible Technology, 2003.
[11] J. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm Intelligence*: Morgan Kaufmann Publishers, 2001.
[12] J. Tillett, R. Rao, F. Sahin, and T. M. Rao, "Particle swarm optimization for the clustering of wireless sensors," presented at SPIE, Orlando, FL, 2003.
[13] J. Tillett, R. Rao, F. Sahin, and T. Rao, "A distributed evolutionary algorithmic approach to the least-cost connected constrained sub-graph and power control problem," presented at SPIE Security and Defense Symposium, Orlando, FL, 2004.

Table 6 – Comparison of DPSO and heuristic approaches using power control

| DPSO approximation of Voronoi radius | Heuristic approximation of Voronoi radius |
|---|---|
|  |  |
|  |  |
|  |  |