An invited paper at the American Nuclear Society (ANS) 10th International Conference on Robotics and Remote Systems for Hazardous Environments, Gainsville, FL.

Swarming Pattern Detection in Sensor and Robot Networks

H. Van Dyke Parunak[1], Sven A. Brueckner[1], James Odell[2]

[1]*Altarum Institute: 3520 Green Court, Suite 300, Ann Arbor, MI 48105-1579, {van.parunak, sven.brueckner}@altarum.org*
[2]*James Odell Associates: 3646 West Huron River Drive, Ann Arbor, MI 48103-9489, email@jamesodell.com*

Abstract – *Monitoring for radiation (e.g., leak detection or finding smuggled materials) requires managing a dynamic spatio-temporal configuration of sensors. One promising approach is to combine fixed sensors with sensors on robots, and endow the population with the ability to configure themselves and coordinate their actions to create and maintain the required sensor configuration. This paper describes some scenarios where such a capability would be useful, identifies technical issues that need to be addressed, suggests general principles and techniques for dealing with such scenarios, and describes a specific example that we have constructed and tested in a simulation environment.*

I INTRODUCTION

A number of nuclear surveillance scenarios can benefit from a swarm of coordinated sensors. A dynamic scenario may require that the distribution of sensors and the roles that they play change over time. Thus at least some of the sensors need to be mobile, and coordination mechanisms must adjust the configuration of the team dynamically in response to changing requirements.

One approach to multi-agent coordination is swarming, "useful self-organization of multiple entities through local interactions."[1] We have developed a number of applications of swarming to problems as diverse as UAV navigation,[2] pattern recognition in sensor networks,[3] and searching for information in immense bodies of textual data.[4] This paper illustrates the applicability of these techniques to problems of robotic surveillance for radioactive materials. Section II describes several application scenarios and identifies requirements that swarming sensors must satisfy. Section III discusses principles and techniques that we have found helpful for such problems. Section IV outlines the application of these techniques to a particular problem. Section V concludes.

II PROBLEM DEFINITION

A number of nuclear surveillance scenarios can benefit from a swarm of coordinated sensors. These include:

Facility monitoring.—Facilities that manipulate nuclear material (including power stations, research reactors, and weapons dumps) must be monitored for leakage. Potential leakage sites may be widely distributed throughout the facility. Once a leak is detected, the dispersion of radioactive material must be monitored, and a protective perimeter established to avoid inadvertent human contact.

Smuggling detection.—Dockyards, railroad marshalling yards, and warehouses present the challenge of monitoring large areas for suspected materials. Surveillance must combine wide-spread application of highly generic sensors with subsequent validation with more sophisticated sensors to weed out false positives.

Release management.—The possibility of release of radioactive material in an uncontrolled area (as in the case of a terrorist attack) makes it desirable to maintain continuous surveillance to provide early detection and localization of the source, monitoring of its dispersion, and establishment of a protective perimeter.

These scenarios, and others like them, have a number of common requirements that swarms of autonomous sensors can address.

- Sensory input is needed from different spatial locations under tight temporal constraints.
- While some locations may be known in advance and monitored with fixed sensors, others will change over time, requiring at least some sensors to be carried on robots. In addition, mobile sensors reduce the number of sensors needed to cover a given area.
- The roles played by sensors and their mobility platforms often need to change in response to dynamic events.

Because we envision a heterogeneous population of sensors, we use a variety of terms in describing our mechanisms. The generic entity is "sensor," which may be stationary or mobile. A mobile sensor is mounted on a "mobility platform," and the entire entity is thus a "robot." Either class of sensor may be termed an "agent" to emphasize its autonomous computational abilities rather than its specific sensory function.

III GENERAL PRINCIPLES AND TECHNIQUES

Three classes of principles and techniques are useful in addressing problems of this nature. The first class concerns the relation between individual agents and the groups of which they are a part, and falls under the general rubric of "roles." The second concerns mechanisms for optimizing such systems in the face of resource constraints. The third is the broad area of mechanisms inspired by natural systems.

III.A Issues in Team and Role Coordination

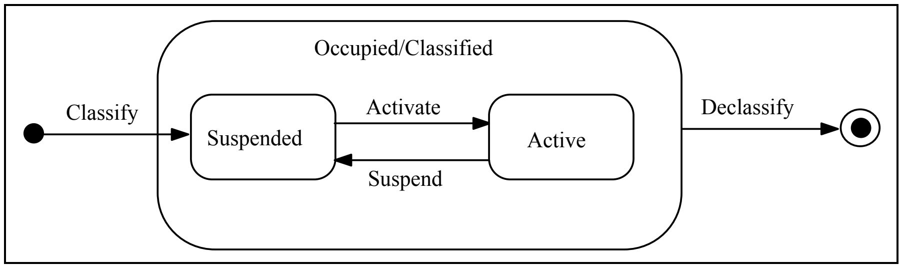

Fig. 1. Statechart depicting some of the permitted states and transitions of an agent in a role.

Effective modeling and design of agents in emergent swarms is greatly facilitated by identifying distinct "roles," or patterns of behavior, that agents can "play" in different mission settings.[5] A role is a class that defines a normative behavioral repertoire of an agent. Roles provide both the building blocks for agent social systems and the requirements by which agents interact. Each agent is linked to other agents by the roles it plays by virtue of the system's functional requirements—which are based on the expectations that the system has of the agent. The static semantics of roles, role formation and configuration, and the dynamic interactions among roles have been examined closely in recent years,[5-9] and an initial axiomatization has been proposed.[6] However, little work has been done on formalizing the temporal aspects of dynamic role assignment. Role modelers refer only informally to actions such as "taking on a role," "playing a role," "changing roles," and "leaving roles." The ambiguities inherent in these terms pose difficulties for applications such as ours, in which dynamic role change is a pervasive feature of the system's behavior.

To understand these issues better, we distinguish two aspects to a change in role, summarized in Fig. 1 and discussed more fully elsewhere.[10] Role classification gives an agent the methods necessary to execute the behaviors in a role). Role activation captures the sense that an agent is currently executing in a role.

III.A.1 Dynamic Classification

Dynamic classification refers to the ability to change the classification of an entity. Consistent with the proposed axiomatization,[6] we insist that each agent have at least one role at all times. Dynamic classification deals with adding additional roles or removing roles beyond the minimum of one. This requirement is analogous with the notion that every human must play the "person" role, whatever other roles they may have. In the case of humans, this minimal role persists throughout the agent's life. It is conceivable that an artificial agent might begin with the minimal role A, add role B, then remove role A, leaving it with the minimal role B. Whether such a fundamental redefinition of the agent is possible will depend on such features as physical equipment associated with the agent and the nature of the platforms on which the agent can run. An alternative approach is to define a basic role AgentId that belongs to every agent, whatever other roles it may play. (Id in AgentId recalls the Freudian notion of primal basic urges, not "Identity.") Having AgentId as a role is a controversial point. However, elsewhere[11] we have defined role as a class that defines a normative behavioral repertoire of an agent. The basic class AgentId defines the normative behavioral repertoire for agenthood.

To become an instance of a given role, the agent is classified as an instance of, or occupies, that role. Once classified, the agent occupies the new role and possesses all of its features. In the opposite process, if an agent is declassified, it is removed as an instance of a particular role—and no longer occupies the role nor possesses features unique to that role. An agent is said to be reclassified when it is both declassified in one role and classified as another. Agent instantiation and deletion are limiting cases of changes in classification, and we describe their consequences at the role level with create and delete operators. TABLE I summarizes roles held by an agent before and after each of these operators.

TABLE I: Operators for Dynamic Classification

Operator	Pre-state	Post-state
Classify	A and not B	A and B
Declassify	A and B	A and not B
Reclassify	A	B
Create	Null	A
Terminate	A	Null

III.A.2 Dynamic Activation

In addition to changing roles over time (dynamic classification), an agent may have multiple roles that apply to it at any one moment, a condition that we describe as *multiple classification* (not to be confused with multiple inheritance). Role *activation* captures the intuition that an agent may hold multiple roles concurrently while not actively executing them at the same instant.

Formalizing such a notion of "activity" is problematic. In some sense, even a quiescent agent that is waiting for a message or some signal to awaken could be considered active in its role, because alertness can be thought of an activity. UML

2.0[12] offers a useful refinement by distinguishing between user-defined actions (which are represented explicitly in sequence diagrams and activity diagrams) and fundamental system actions such as i/o, invocation, and data flow (which are not represented as actions in these diagrams). In UML, each activation, or *execution occurrence*, has some duration and is bounded by a start and stop point. We propose to take advantage of this refinement in the following unification:

- *We adopt the UML 2.0 definition of action.* Any unit of behavior that has started and has not yet ended is considered "active." Otherwise it is "inactive."
- *We use the basic role of* AgentId *to specify primitive behavior.* Behaviors such as controlling, handling data flows, and waiting for messages and signals are "primitive" actions that all entities must possess to be agents. Therefore, any entity playing the role of AgentId can exhibit this basic behavior, deferring "higher-level" behavior to user-specified actions in more specialized roles. Furthermore, these basic behaviors are themselves actions. For example, actions that support listening for messages and signals, *by definition,* begin the moment an entity is classified an AgentId and cease when the entity is no longer an AgentId.
- *We consider roles other than* AgentId *to be active only when their user-defined actions are active.* Activity of primitive actions is attributed to the concurrently executing AgentId role, not to the user-specified role.

Dynamic activation involves the operators *activate*, *suspend*, and *shift*. TABLE II summarizes the role assignments as affected by these operators.

TABLE II: Operators for Dynamic Activation

	Prestate		Poststate	
Operator	**Active**	**Suspended**	**Active**	**Suspended**
Activate	A	B	A and B	
Suspend	A and B		A	B
Shift	A	B	B	A

III.B Resource Constrained Local Optimization

In the execution of a particular mission, we deploy many simple agents in a mission swarm and task the swarm (not the individual agent) with the mission goal (e.g., mapping a leak with required resolution). Once tasked, the agents in the swarm coordinate their individual activities (see Team and Role Coordination) to achieve the mission goal.

Especially if a mission takes a long time to complete, the swarm needs to be able to judge the current quality of mission achievement and adjust its execution accordingly. In the case of distributed mapping of a leak, a single sensor may have detected the presence of the leak, and now the swarm needs to re-configure the formation to determine the leak's extent and monitor its spread. Fig. 2 shows the resulting closed adjustment loop.

Adapting the swarm's mission execution to the quality of the mission goal achievement requires a) that the swarm is able to *perceive* the current quality, and b) that the swarm *knows how* to reconfigure to improve this quality. Conventional optimization approaches assign this decision to a central computer, but such an architecture does not scale easily, requires high bandwidth so that all sensors can communicate with the central station, and is vulnerable to failure or attack of the central processor. In the absence of a "swarm" entity, these two issues immediately translate to the need for the individual agent to know whether and how it needs to change its behavior to improve the overall (global) system performance.

In the following paragraphs, we contrast three approaches to the re-configuration problem: implicit local models, fitness evaluation, and quality evaluation. We review the applicability of these approaches with respect to the effort involved in the specification of the knowledge of the individual sensor and the processing and communication effort that arises in the execution.

All three cases assume local autonomous control by the sensor (no central controller). The re-configuration process is iterative: each sensor runs through multiple cycles of perception (receiving input from other sensors), planning (selecting a new role and, if mobile, a new location), and execution (assuming the new role and/or approaching the selected location). An individual sensor's decisions about role and location assume that if no other sensor were to change, the resulting new formation would better satisfy the mission objectives. Other sensors do change, and invalidate this assumption, but the process repeats and iteratively converges.

III.B.1 Implicit Local Model

An implicit local model is a (numerical) function given to a mobile sensor that takes as an input the state of the swarm (locations) as perceived by the sensor and returns the new location that the sensor should assume.

Consider for example mapping a leak. A formation

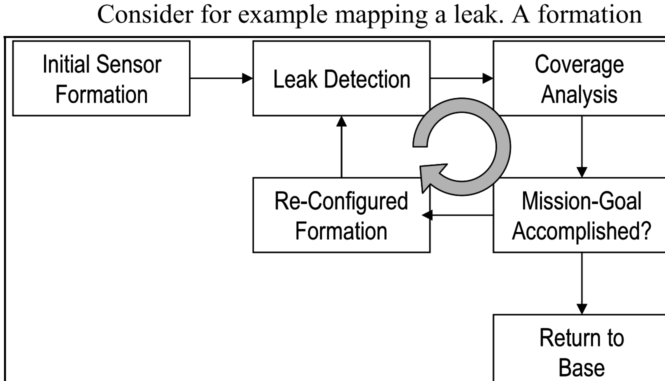

Fig. 2. A swarm adjusts its execution to meet the mission goal.

that is denser around the edges of the leak than near the source gives good definition. So, an implicit local model for the sensor would be a function that computes from the current formation a location for the sensor that would increase sensor density around the limits of the leak.

To provide the sensor with an implicit local model, we need to understand the characteristics of those emergent sensor patterns that lead to good performance and we need to craft the local transformation function that determines the best change for one sensor given the overall arrangement. Furthermore, since the swarm reconfiguration emerges from the decisions of the individual sensors without any representation of the global state or goals, the design process must include extensive validation of the implicit model. Thus, the approach requires rather complex knowledge engineering before deployment. In compensation, the cost incurred by the approach during execution is minimal, since the sensor only needs to execute the transition function once to complete the planning process.

III.B.2 Fitness Evaluation

If the construction of a function that generates the new location directly is not feasible, a local search approach may be followed. Our second approach constructs a fitness function that translates a given state of the swarm (sensor locations) into a fitness value (e.g., a number in [0,1]). Thus, the individual sensor may perform a search for a nearby peak in the fitness landscape that is spanned by the (virtual) variation of its location in the overall swarm arrangement.

In the case of mapping a leak, a useful fitness function might evaluate how appropriate a given set of sensor locations is, given, their current feedback. The individual sensor would then seek to improve the quality of the map by assuming a new location that results in a more appropriate distribution.

The reasoning process of the individual sensor is still partially implicit. Though the use of the fitness function now explicitly analyses the global state of the swarm, it still includes the implicit assumption that a particular arrangement will result in good map quality. So, from a knowledge engineering perspective, the designer still has to understand the link between the state of the swarm and the quality of the mission performance.

The use of a fitness evaluation of possible swarm states results in a higher computational effort during the execution of the mission. The sensor must generate and evaluate possible formations that only vary its own location relative to the currently perceived arrangement of the other sensors. Once it finds a formation with a sufficiently improved fitness, it will take its own location in this formation as the goal for its subsequent relocation process.

III.B.3 Quality Evaluation

Specifying a metric that explicitly measures the quality of mission achievement for a given state of the swarm is the most explicit approach. Instead of assuming that a certain state results in a certain quality, this approach translates a given state into an expected outcome and then explicitly measures the quality of the result.

In the case of leak mapping, we would construct a simulator that translates a sensor formation into the expected map. Then we would measure, for instance, the contrast or spatial resolution that would be achieved.

The knowledge engineering process for this approach is relatively simple and completely driven by the final goal of the respective mission. The design includes modeling the process of mission execution (the map acquisition and sensor processing) and the specification of mission quality metrics.

Depending on the computational cost of predicting the outcome for a particular swarm state, the execution of the mission optimization process can be very expensive. As in the case of the fitness evaluation of a given state, each sensor performs a search across possible formations, but rather than applying a single function, the sensor must first estimate the mission outcome and then apply the quality metric.

III.B.4 Summary of Optimization Methods

If we deploy swarms of many simple sensors to perform an extended mission, we need to endow the swarm with the ability to adapt its mission execution based on the quality of the quality of the goal achievement. The autonomy of the sensors and the limited availability of processing and communications resources as well as the uncertainty and noise in the interactions with the physical environment (sensing and acting) require that the individual sensor change its behavior locally to improve the

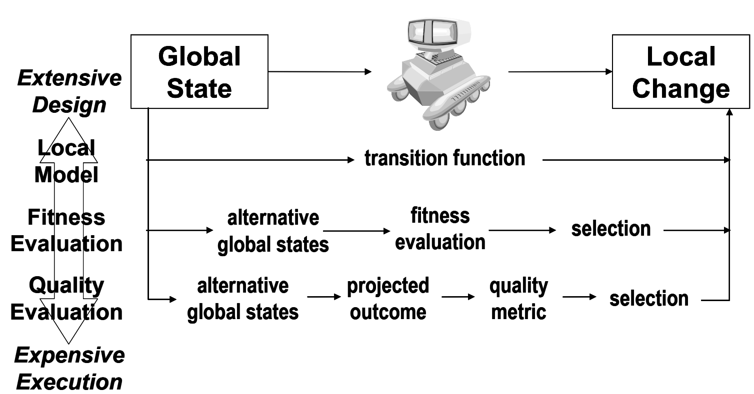

Fig. 3. Three approaches to mission quality optimization.

overall mission performance.

Fig. 3 summarizes our three approaches to local optimization of the mission performance. They vary in the degree to which the individual sensor must be aware of the current mission quality and means to improve it. Endowing the sensor with an implicit local model results in the fastest and simplest execution of the swarm, but requires a significant effort in designing and validating the model. Using a fitness function that evaluates the quality of a given formation with respect to the implicitly assumed mission performance relieves the designer of the burden of constructing a complete model since it applies a heuristic search on potential alternatives, but still requires off-line specification of what a "good" swarm state would be. The simulation of the outcome produced by alternative states restricts the knowledge engineering to the description of the operation of the system, but then requires the sensor to figure out the link between the process and appropriate optimization strategies.

The specific characteristics of the three proposed approaches suggest the following simulation-based transition path. In the first stage, we apply our knowledge of the general process to optimize the mission performance based on the quality evaluation approach (simulate outcome and measure quality). We then perform extensive simulations of the intended mission and characterize the swarm states that result in good mission performance (e.g., preferential allocation of sensors to the boundary of a leak). Based on this characterization, we construct a fitness function that measures the degree to which a given state deviates from the description of a "good" state. We replace the prediction and quality metric with the fitness function and validate its performance in the simulation. Finally, we analyze the nature of the relocation decisions that the individual sensor makes and construct and evaluate the implicit local model from which the "good" global states emerge. At this point, the computational effort required for the execution of the mission optimization is sufficiently small to deploy the decision logic onto a real resource-constrained sensor.

III.C Inspiration from Natural Systems

Our approach to autonomous coordination among multiple entities is based on principles from biological communities, which we have outlined elsewhere.[13] TABLE III gives examples of the kinds of behavior that these techniques can support. Broadly speaking, these techniques achieve self-organization in multi-agent systems by way of local interactions.

TABLE III: Natural Examples of Swarming

Swarming Behavior	Entities
Pattern Generation	Bacteria, Slime Mold
Path Formation	Ants
Nest Sorting	Ants
Cooperative Transport	Ants
Food Source Selection	Ants, Bees
Thermoregulation	Bees
Task Allocation	Wasps
Hive Construction	Bees, Wasps, Hornets, Termites
Synchronization	Fireflies
Feeding Aggregation	Bark Beetles
Web Construction	Spiders
Schooling	Fish
Flocking	Birds
Prey Surrounding	Wolves

Elsewhere, we have reported how these techniques can be applied to such practical problems as coordinating manufacturing operations,[14] planning paths and deconflicting airspace for UAV's,[2] and recognizing patterns in a distributed sensor network without central processing,[3] and compared our approach (based on agent interactions through a shared environment) with others.[1]

IV A SPATIAL COORDINATION EXAMPLE

As an example, we describe a swarming architecture to do leak detection, mapping, and perimeter control for a nuclear facility. In this scenario, the swarm must achieve four objectives that require different behaviors on the part of individual UAV's.

In **searching**, it must effectively cover a large search space. Fixed sensors are installed at critical locations, but there are not enough fixed sensors to monitor all potential leak locations (e.g., long pipe runs). Thus mobile sensors supplement the fixed sensor network. A sensor in search mode compares its readings with a threshold to determine whether it should report a potential leak or not. This threshold may be determined statically, but more robust results can be obtained if the sensor sets its threshold based on reports from its nearest neighbors.[15]

When one sensor detects a leak, it announces its location, and sensors that receive this announcement begin **mapping.** In this phase, mobile sensors seek to distribute themselves along the boundary of the leak by comparing readings from their neighbors and moving to a location where neighbors on one side sense radiation while neighbors on the other side do not. Stationary sensors perform additional processing and communication to determine whether they are internal to, on the boundary of, or external to the contaminated area.

Once a leak has been detected and mapped, a subset of sensors assume the role of **sentries**. Mobile sensors that move back from the perimeter join fixed sensors that are a minimal distance from the perimeter of the leak. Sentry sensors adjust their location in response to changing positions reported by boundary sensors, and also activate warning devices to alert humans to the danger.

In addition, individual mobile sensors require periodic refueling or other **maintenance**, and the swarm must ensure that individual vehicle requirements are met without compromising the ability of the overall swarm to continue functioning.

We have defined and implemented swarm coordination mechanisms for this example. Fig. 4 shows an annotated nominal screenshot. Mobile sensors that currently fill the search role are shown as solid triangles, those on their way to refueling are open triangles, those that map the current perimeter of a contaminated area are open lozenges, and sentries are solid lozenges. Solid circles are fixed sensors.

For each of the four roles in this example, we design behaviors of individual mobile sensors to achieve the overall (system-level) mission objectives. These individual behaviors may include direct coordination with other sensors, indirect coordination, or no coordination at all. In the following, we sketch the behaviors of mobile sensors as they fill the respective roles.

IV.A Searching

The goal of the mobile sensors in the searching role is to cover the space where leaks are possible. Furthermore, the search should be structured to cover areas with higher leakage probability more often.

In our example, we assume a simplistic two-dimensional search space in which the probability of a leak decreases proportional to the distance from the base of the mobile sensors ("The Facility"). (Real applications have more complex topologies.) We define a set of paths (trajectories through space) that searching sensors may occupy. These paths are circles around the base identified by their radius.

Mobile sensors that decide to assume the searching role select one of these paths and start to traverse it. Assuming for now that mobile sensors are capable of communicating with their base station at the central facility, we designate the base as the mediator of the path selection. It is the goal of the sensors to distribute their assignments to paths as evenly as possible. Thus, a sensor assuming the search role will choose the path with the lowest number of sensors already assigned. This information is provided by the base, which tracks the assignments. Alternatively, sensors could broadcast their decisions and thus track each other's choices in a decentralized fashion.

With an even distribution of sensors to radial paths and assuming that the speed of the sensor does not depend on the chosen path, areas farther away from the base are revisited less often than those closer by. Thus, the swarm focuses its attention on higher-probability leakage areas. Other probability distributions may require more explicit reasoning in the assignment process.

The sensors furthermore seek to optimize the search process by coordinating their progress along the chosen path. Depending on how many sensors have signed up for a path, there is an optimal distance between the sensors

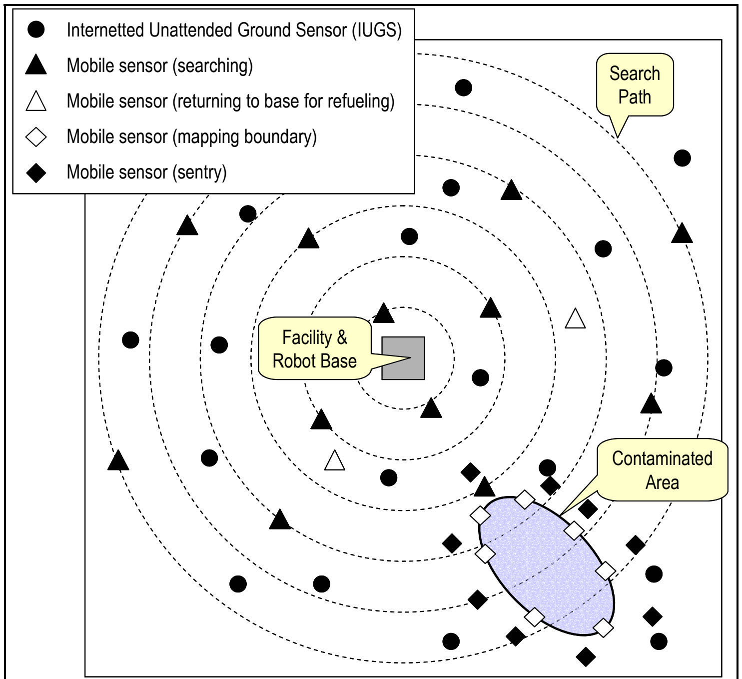

Fig. 4: Individualistic search strategy

that should be achieved to avoid rapid revisits with large gaps in between. The sensors communicate (through the base) their current position along the path and adjust their individual progress based on the observed progress of other sensors on the same path. Again, this distribution could be achieved by distributed means by propagating a message among sensors on the same search path.

A sensor that detects a potential leak dynamically recruits a mapping team and a sentry team. Agents bid for a role in the team based on the match of the sensor's capabilities with the role's requirements (hard constraint) and the current distance of the sensor to the detected target (soft constraint).

Once roles are assigned, mobile team members adjust their locations in response to sensory feedback from their nearby neighbors, so that mapping agents assume positions on the boundary of the leak while sentry agents stand back from them and activate warning mechanisms. Once a human overseer acknowledges the detection and initiates recovery procedures, the team disbands and the agents resume their search behavior.

IV.B Mapping

In a dynamically announced mapping mission mobile sensors seek to position themselves near the edge of a contaminated area in a way that maximizes the spread of sensors along this edge. The individual sensor perceives the "edge" as a threshold on its own sensor readings dynamically adjusted by readings of nearby (stationary and mobile) sensors.

The mobile sensors coordinate their positions through a dynamic force model. Each sensor repeatedly recomputes its preferred location, calculating an attractive force towards sensors at the edge of the area and repulsive force components away from other nearby sensors. It then attempts to move towards this preferred location.

The integration of the individual movement decisions across time with the physical extent of the contamination and the arrangement of the static sensors leads to the emergence of the desired arrangement of mobile sensors along the edge of the contaminated area.

IV.C Guarding (Sentry)

While the sensors that form the mapping team coordinate their positions with respect to the edge of the contaminated area and the positions of other mapping teammates, the sentry team coordinates its position relative to the mapping team.

Sentries also use a force model to determine their individual motion. Any mobile sensor inside the contaminated area is repelled by the center of gravity (average position) of the mapping team. A mobile sensor outside of the contaminated area is repelled from nearby members of the mapping team, as long as its distance from the closest mapping team member is smaller that the required safety distance of the sentries from the contaminated area. Otherwise it is attracted to these nearby members.

IV.D Maintenance

To accomplish maintenance, each time a mobile sensor reviews its current role, it evaluates an exponential probability distribution over its fuel level. Full sensors almost never return for refueling, while almost empty ones are highly likely to return. The stochastic nature of the decision breaks the symmetry among sensors with similar fuel levels, and the swarm stabilizes in a state where a fixed proportion is engaged in maintenance at any time, assuming similar fuel consumption on the part of all sensors.

IV.E Discussion

This example illustrates the utility of the three principles that we discussed earlier.

IV.E.1 Roles Analysis

A sensor's shift among search, mapping, sentry, and maintenance is a showcase example of the usefulness of role-oriented design for high-level agent specifications. The code for each sensor includes all three roles, so there is no dynamic classification. Dynamic activation is invoked in the form of the *shift* operator between distinct roles.The *shift* from search to mapping or guarding takes place only after an agent wins the bidding process. The return *shift* from mapping or guarding to search is unilateral, after a human overseer releases the team.

IV.E.2 Optimization

Our approach uses an implicit local model for optimization. Each sensor's algorithm is based on our knowledge that the boundary of a leak is defined by a transition from high to low sensor readings. The agents do not know anything about the quality of their mission performance, or how to translate such knowledge into waypoints. This level of optimization economizes run-time execution at the expense of up-front engineering to determine the parameters required for effective performance. Such implicit models are highly desirable for efficient deployment on sensors, especially in support of small inexpensive platforms, and methods for their systematic development are an important topic for research.

IV.E.3 Natural Systems Analogs

The use of virtual attractive and repulsive forces in mapping and sentry coordination is a direct application of environmental coordination mechanisms (sematectonic stigmergy) inspired by natural systems, and is inspired by an algorithm observed in wolf packs.[13] Stigmergic systems are of particular value in applications that are discrete, distributed, decentralized, and dynamic, and all four characteristics are important for our application. These algorithms are appropriate systems of *discrete* entities, such as a swarm of sensors. Virtual forces enable *distribution* of system knowledge over the swarm, since each mobile sensor may maintain a local map of its vicinity, and *decentralization*, since all sensors are peers in the coordination algorithm, making the system robust to loss of any individual platform. Perhaps the most important benefit of stigmergic techniques in this application is their support for *dynamic* environments. A given sensor's environment (which includes the other members of the swarm) is constantly changing, and coordination mechanisms based on traditional knowledge bases face a huge challenge in maintaining consistency. Virtual forces are constantly recomputed, disposing any information that is not being reinforced by current observation, and thus automatically discard obsolete information.

V CONCLUSION

An important application for swarming coordination is coordinated sensing tasks. We have developed a suite of design principles and algorithmic approaches to this coordination, and demonstrated their effectiveness in software simulations.[16]

ACKNOWLEDGMENTS

This work is supported in part by DARPA, contract F30602-02-C-0196 to Altarum, under DARPA PM Vijay Raghavan. The views and conclusions in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

REFERENCES

1. H.V.D. PARUNAK, "Making Swarming Happen," In *Proceedings of Swarming and Network-Enabled C4ISR*, Tysons Corner, VA, ASD C3I, 2003. http://www.erim.org/~vparunak/MSH03.pdf.
2. H.V.D. PARUNAK, M. PURCELL, and R. O'CONNELL, "Digital Pheromones for Autonomous Coordination of Swarming UAV's," In *Proceedings of First AIAA Unmanned Aerospace Vehicles, Systems,Technologies, and Operations Conference*, Norfolk, VA, AIAA, 2002. www.erim.org/~vparunak/AIAA02.pdf.
3. S.A. BRUECKNER and H.V.D. PARUNAK, "Swarming Agents for Distributed Pattern Detection and Classification," In *Proceedings of Workshop on Ubiquitous Computing, AAMAS 2002*, Bologna, Italy, 2002. http://www.erim.org/~vparunak/PatternDetection01.pdf.
4. H.V.D. PARUNAK, P. WEINSTEIN, S. BRUECKNER, and J. SAUTER, "Hybrid Stigmergy for Information Extraction," In *Proceedings of Mathematics and Algorithms of Social Insects (MASI03)*, Atlanta, GA, 2003.
5. H.V.D. PARUNAK and J. ODELL, "Representing Social Structures in UML," In M. WOOLDRIDGE, G. WEISS, and P. CIANCARINI, Editors, *Agent-Oriented Software Engineering II,*, *Lecture Notes on Computer Science*, pages 1-16. Springer, Berlin, 2002. http://www.erim.org/~vparunak/AOSE2001.pdf.
6. J. FERBER, O. GUTKNECHT, and F. MICHEL, "Agent/Group/Roles: Simulating with Organizations," In *Proceedings of Fourth International Workshop on Agent-Based Simulation (ABS03)*, Montpellier, France, 2003.
7. C. CASTELFRANCHI, "Engineering Social Order," In, *Engineering Societies in the Agent World*, pages 1-18. Springer, Berlin, 2000.
8. M. DASTANI, V. DIGNUM, and F. DIGNUM, " Role Assignment in Open Agent Societies," In *Proceedings of Second International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS'03)*, Melbourne, Australia, pages (forthcoming), 2003.
9. J. ODELL, H.V.D. PARUNAK, and B. BAUER, "Representing Agent Interaction Protocols in UML," In *Proceedings of Agent-Oriented Software Engineering*, 22nd International Conference on Software Engineering, pages 121-140, Springer, 2000. http://www.jamesodell.com/Rep_Agent_Protocols.pdf.
10. J.J. ODELL, H.V.D. PARUNAK, S. BRUECKNER, and J. SAUTER, "Temporal Aspects of Dynamic Role Assignment," In *Proceedings of Workshop on Agent-Oriented Software Engineering (AOSE03) at AAMAS03*, Melbourne, AU, pages (submitted), Springer, 2003. http://www.erim.org/~vparunak/TemporalAspects03.pdf.
11. J. ODELL, H.V.D. PARUNAK, and M. FLEISCHER, "The Role of Roles in Designing Effective Agent Organizations.," In A.F. GARCIA and ET AL., Editors, *Software Engineering for Large-Scale Multi-Agent Systems*, Springer-Verlag, Berlin, 2003.
12. OMG, "UML," 2003, Web Site. http://www.omg.org/uml/.
13. H.V.D. PARUNAK, "'Go to the Ant': Engineering Principles from Natural Agent Systems," *Annals of Operations Research*, 75:69-101, 1997. http://www.erim.org/~vparunak/gotoant.pdf.
14. S. BRUECKNER, *Return from the Ant: Synthetic Ecosystems for Manufacturing Control,* Dr.rer.nat. Thesis at Humboldt University Berlin, Department of Computer Science, 2000. http://dochost.rz.hu-berlin.de/dissertationen/brueckner-sven-2000-06-21/PDF/Brueckner.pdf.
15. S. BRUECKNER and H.V.D. PARUNAK, "Decentralized Detection, Location, and Tracking with Distributed Sensors." USA Patent # (provisional), 2002.
16. H.V.D. PARUNAK and S. BRUECKNER, "Swarming Coordination of Multiple UAV's for Collaborative Sensing," In *Proceedings of Second AIAA "Unmanned Unlimited" Systems, Technologies, and Operations Conference*, San Diego, CA, AIAA, 2003. http://www.erim.org/~vparunak/AIAA03.pdf.