Decision Making in a Distributed Sensor Network

Satu Elisa Schaeffer[†] Jonathan P. Clemens Patrick Hamilton
elisa.schaeffer@hut.fi jonathan.p.clemens@intel.com patrick.hamilton@charter.net

August 6, 2004

Abstract

Given a binary output sensor network of a particular density and a random distribution, we examine the effects of simple malfunctioning sensor nodes on the ability of the network to reach a timely and correct decision. We examine the effects of malfunctioning sensor frequency and distribution, and network decision-making thresholds on the efficacy of the decision-making process. Based on repeated simulations, we are able to give recommendations on how to tune the design parameters of a sensor network to fit a specific application when the cost of false positives and false negatives can be estimated.

1 Introduction

A *sensor network* is a collection of *sensor nodes* that is spread around an area in which a certain phenomenon of interest is expected to take place [2]. Often the phenomenon is some type of a security threat, the presence of which needs to be decided by the network [5]. An example could be a national park, where forest fires are to be avoided — sensor nodes that are capable of detecting e.g. smoke or high temperatures are scattered in the park and the network needs to propagate an alarm to the forest guard in case of a fire. In most cases, the sensor placement is not a carefully designed process but a rather random scattering. Each sensor node is composed of essentially four components [1]:

- the *sensing* unit that makes observations of the environment; in this paper, we limit our attention to binary input sensors,
- the *processing* unit that determines what actions need to be taken; this is commonly a very limited computational device with little memory,
- the *transceiver* unit that receives and broadcasts signals enabling nearby sensor nodes to communicate; usually the range of the broadcast is somewhat limited,
- the *power* unit — essentially a battery — that supplies energy for the other components; in this study, we exclude energy-related issues, solutions to which have been discussed in several papers (see e.g. [6, 8, 12]).

The data communication between sensors is another complex, much studied topic that we leave unattended in this study. We assume that the sensors may have a method of neighborhood detection, but in the cases studied in our experiments (Section 4), it suffices that a sensor is able to detect when a neighboring sensor is broadcasting an alert. In this study, we have no need for a *routing protocol*, as the sensors only communicate to their immediate neighbors; issues of routing are much studied in existing literature (see e.g. [7] and the references therein).

[†]Address for correspondence: Helsinki University of Technology, P.O. Box 5400, FI-02015 HUT, Finland.

The sensor works as a *detector*: given a set of *observables* (the possible broadcasts of the neighboring sensors and the state of the local environment detected by the sensing unit), the sensor must choose between possible *hypotheses* (whether there is an alert condition present or not) [5]. In our restricted scenario, a sensor v calculates a function $f : \{0,1\}^{\deg(v)+1} \to \{0,1\}$ — given the binary output of each of the $\deg(v)$ neighbors and the binary variable sensed from the environment, decide whether to broadcast an alert ($1 \triangleq$ true) or stay silent ($0 \triangleq$ false). There is usually little *a priori* information on the likelihood of an alert, so the computation often relies only on the observations of the sensor itself [5].

Issues of interest, studied in this paper, are the probability that a sensor network correctly detects an alert condition and propagates this information to an outside observer (in our model, represented by a *target area* which the alert signal must reach) and the rate of false alarms [5]. A false alarm may be costly (e.g., sending in the fire brigade when there is no fire), but losing a correct alert is in many cases much more devastating. Finding a balance between these two conflicting goals is an important question in designing a sensor network for a particular application, for which the costs of false negatives and false positives can be estimated to some extent.

2 Modeling Sensor Networks

Very often a sensor network model is built on the following assumptions (generalized from [12]): sensors are deployed on a square area, often a unit square. Each sensor has a communication radius, and usually the circular area within this radius is considered reachable. The network contains at least one *data sink*, i.e., a node or an area to which the observations and/or decisions of the network are communicated. Each sensor is assumed to have a (fixed) lifetime, after which it ceases to operate. Sensors may also suffer damage from the environment or undergo other types of malfunctions [11]; their operation may be either distorted or completely prevented due to such system failures. It is also imaginable that a malicious adversary will attempt to interfere with the sensor network by modifying the sensors or introducing sensors that do not operate as desired by the deployer of the original network.

In this work, we concentrated on a single aspect of sensor networks, namely the decision making in the sense of deciding when to forward an observed alert, given the time-frame and context of the SFI CSSS. This is only one of the numerous pressing research issues related to sensor networks and we hope to provide an approach to aid those designing a sensor network for deployment for a particular task in determining how the sensor should be set up.

3 A Simplified RePast Model

We built a simulation framework for simplified sensor network models using the Java-based RePast simulation tool [9] for agent-based modeling; an executable Java-archive with source code is available at `http://www.tcs.hut.fi/~satu/sensor/`. Table 1 gives a brief discussion of the parameters available for modification in the simulation framework. The edges may be drawn in the visualization, but for even relatively dense networks, this will not be very useful. The alert area A is shown with dark red and the target area T with dark green. Example snapshots of the simulation tool are shown in Figure 1; for extensive runs, a batch interface without visualization was implemented.

4 Signal Propagation Experiments

We used the RePast model for sensor networks to examine decision-making in the network. If an alarm is present (i.e., a non-zero alert area is defined), the network is said to correctly *decide* that there is an alarm if any sensor in the target area will alert. The network is said to make a *false positive* decision if it decides

Agent Count n	This determines the total number n of sensors present in the network. Note that the number of successfully functioning agents will be n minus the number of malfunctioning sensors.
Unit Size	This parameter governs the visual representation of the display, in pixels.
Sensor distribution	The sensors are placed either according to the uniform distribution in $[0,\,1] \subset \mathbf{R}$ or normal distribution (x- and y-coordinates chosen independently). For the normal distribution, the mean and standard deviation can be specified; when sampling from the normal distribution, coordinate pairs that fall outside the unit square $[0,\,1] \times [0,\,1]$ are discarded.
Neighborhood	The neighborhood of a sensor v is determined either by including any sensor that is located within range $r \in \mathbf{R}$ from v's position or by selecting the $k \in \mathbf{Z}$ nearest neighbors using Euclidean distance in the unit square. Both r and k are adjustable parameters. Note that the range-based neighborhoods are symmetrical, but the nearest-neighbor relation is not. A completely connected network can easily be achieved with both settings, choosing either $k = n-1$ or $r = \sqrt{2}$. Neighborhoods are always non-reflexive.
Sensor malfunction	Failures can either be probabilistic, with each sensor having a specified chance to malfunction, or fixed, with a defined number m of failed sensors. The modes of malfunctioning implemented are the following: input always on, input always off, random input, output always on, output always off, random output, and inverse output where the sensor executes the decision computation but reverses the result.
Alert area A	A square area within the unit square, defined by the coordinates of the upper left corner and the lower right corner. Any properly functioning sensor that is located in the alert area will detect input, whereas no other properly functioning sensor will have positive input.
Target area T	A square within the unit square defined similarly to the alert area. These sensors may be assumed to be under the immediate control of the network owner, which prevents malfunctions in them.
Decision computation	Each sensor that detects no alarm itself, but correctly parses its neighbors' alarm broadcasts will make a decision each time step. Two different adjustable rules are implemented: the sensor uses either a percentage or a fixed count of neighbors that need to be in alert state as a trigger to alert itself.

Table 1: The parameters of the simplified RePast model for sensor network decision making.

that there is an alarm when no alert area is present. Similarly, a *false negative* decision occurs when the alert fails to propagate to the target area. The right side of Figure 1 shows an example of a correct alarm decision.

We varied the following parameters: the sensor count n, the communication range r, the alert area A, and the target area T, each *independently*, keeping the others in the following values $n = 200$, $r = 0.2$, $A = (A_u, A_\ell)$ (only present for the true positive/false negative runs) being the upper-left 16th of the unit square, with the upper left corner $A_u = (0,0)$ and the lower right corner $A_\ell = (\frac{1}{4}, \frac{1}{4})$, and $T = (T_u, T_\ell)$ being the low-right 16th, bounded by $T_u = (\frac{3}{4}, \frac{3}{4})$ and $T_\ell = (1,1)$. Failures were not allowed within the target area. As the parameter-variation sets, we used $n \in \{100,\,125,\,150,\,175,\,200,\,225,\,250\}$ sensors, uniformly distributed in the unit square (for methods of ensuring a desired coverage in a sensor network when sensors have a limited battery life, see for example [3]). Neighborhoods were defined by range

$$r \in \{0.05,\,0.1,\,0.15,\,0.2,\,0.25,\,0.3,\,0.35,\,0.4,\,0.45,\,0.5\}.$$

This yields by a simplified calculation the expected number of neighbors per sensor $\bar{k}$, ignoring boundary conditions and hence deriving an upper bound. The probability that a sensor is placed inside a fixed circle of radius r that fits completely inside the unit square is πr^2. As the sensors are placed independently

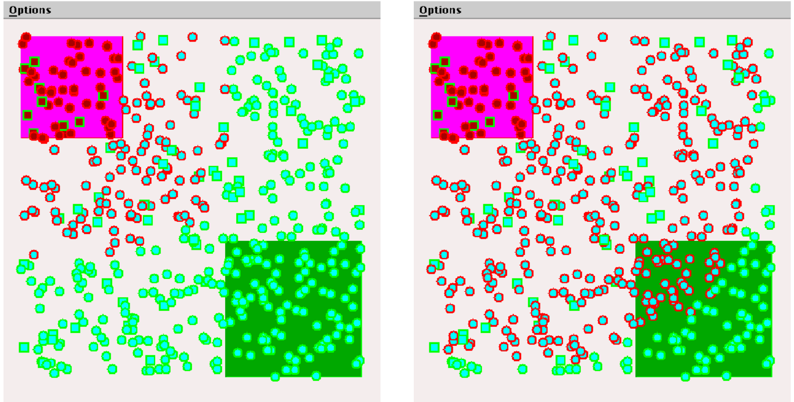

Figure 1: On the left, an example snapshot of the sensor network simulation visualization with a true alarm not yet reaching the target area; on the right, an example of a true alarm causing a correct decision at the target area. Correctly functioning sensors are visualized as circles and malfunctioning ones as squares. The inside of the sensor is red (darker) if it senses an alarm in the environment and green (lighter) if it does not. The border color indicates the output state of the sensor: it is red if the sensor is broadcasting an alert and green otherwise.

and uniformly at random, the number of sensors within such a circle is binomially distributed. There are $n-1$ possible neighbors, each having a πr^2 probability of being located within range, which gives $\bar{k} = \lceil (n-1)\pi r^2 \rceil$. See Table 2 for the estimates on different setups — the table also lists the combinations of n and r studied. We also ran experiments where the alert size was varied: $A_u = (0,0)$ and

$$A_\ell \in \{(0.1, 0.1), (0.15, 0.15), (0.2, 0.2), (0.3, 0.3), (0.35, 0.35), (0.4, 0.4), (0.45, 0.45), (0.5, 0.5)\},$$

keeping $n = 200$ and $r = 0.2$. Similarly, we varied the target area such that $T_\ell = (1,1)$ and

$$T_u \in \{(0.5, 0.5), (0.55, 0.55), (0.6, 0.6), (0.65, 0.65), (0.7, 0.7), (0.8, 0.8), (0.85, 0.85), (0.9, 0.9)\}.$$

The adjusted parameters for each (n, r, A, T)-tuple were the malfunction probabilities $p_t \in [0,\ 0.5]$ (for the true alarms; increments of 0.05), $p_f \in [0,\ 0.05]$ (for the false alarms; increments of 0.005), and the number of neighbors $\gamma \in [1,\ 5]$ used to trigger an alert in a sensor. We ran two sets of tests for both signal propagation settings (the true and the false alarms), altering the synchronization of the network. In the first set, all sensors fire simultaneously and update their state afterward, whereas the second set updates sensors in random order, independently from step to step. The results of the synchronized update experiments and the random update order experiments were nearly identical with respect to the number of false positives/negatives, which leads us to conclude that the synchronization does not affect the decision-making process, although it may and presumably does affect the time in which the signal propagates to the target area.

In the false negative experiments, malfunctioning sensors always had output off. The false positive experiments use malfunctioning sensors that always have input on. The model was allowed to run for 20 steps and then cut off; according to our observations, the signal either reaches the target area in 20 steps or stalls outside of it in almost all cases. A thousand repetitions were taken for each parameter combination examined.

Table 2: Estimated average degree $\bar{k} = \lceil (n-1)\pi r^2 \rceil$ for each of the sensor network setups studied.

n	r	k	n	r	k
100	0.2	13	200	0.05	2
125	0.2	16	200	0.1	7
150	0.2	19	200	0.15	15
175	0.2	22	200	0.25	40
200	0.2	26	200	0.3	57
225	0.2	29	200	0.35	77
250	0.2	32	200	0.4	101
200	0.45	127	200	0.5	157

5 Results

In this section we present and discuss some plots of the experimental results, examining the selection of the alarm trigger threshold γ under different network setups. First we discuss the effect of varying the total sensor count in the unit square, second the variation of communication range, and third, the effects of varying the alert and target area sizes are reported.

5.1 Sensor count variation experiments

We studied the optimal number of neighbors to be used as a trigger for a sensor node to broadcast an alert under possible sensor malfunctions in the network with different failure probabilities. See Figure 4 for a comparison over the (200, 0.2) and (150, 0.2) runs.

If there are either no ($p_t = 0$) or many failures ($p_t \geq 0.45$), trusting only one neighbor to signal is sufficient in propagating a true alert in a (200, 0.2) network, but this is sensitive to false alarms. For most other failure rates, using $\gamma = 4$ seems to provide the optimal decision-making behavior for $n \in \{150, 200\}$; only for $p_t = 0.4$, $\gamma = 3$ excels. For $p_t = 0.4$ and $p_f = 0.04$, the network starts being close to useless: at $p_t = 0.45$ and $p_f = 0.045$, it gives bad data more often than correct data.

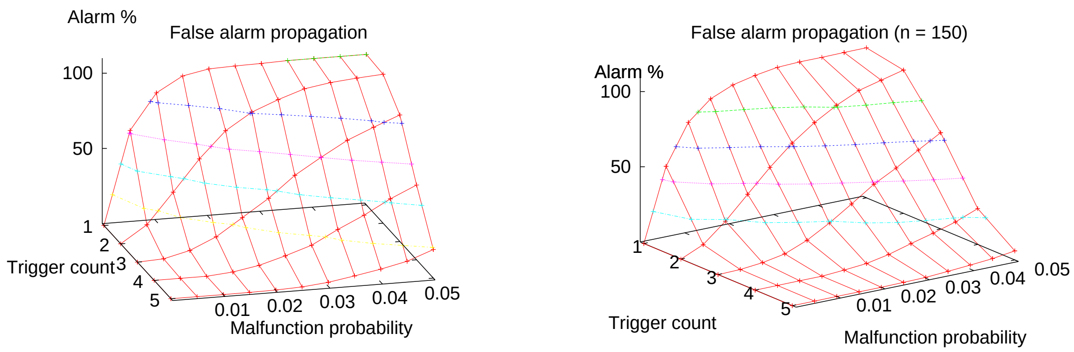

Figure 2: The percentage out of 1,000 runs for $n = 200$ (on the left) and $n = 150$ (on the right) that propagated the alert decision to the target area in a synchronized sensor network under real alarm in the alert area using one to five neighbors as the decision criteria and having a percentage of malfunctioning sensors that always output false and may cause a false negative.

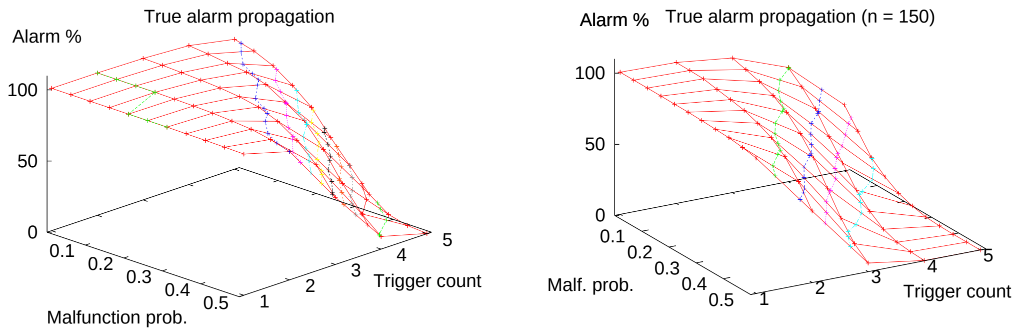

Figure 3: The percentage out of 1,000 runs for $n = 200$ (on the left) and $n = 150$ (on the right) that propagated the alert decision to the target area in a synchronized sensor network under no alert condition using one to five neighbors as the decision criteria and having a percentage of malfunctioning sensors that always output true and hence may cause a false positive.

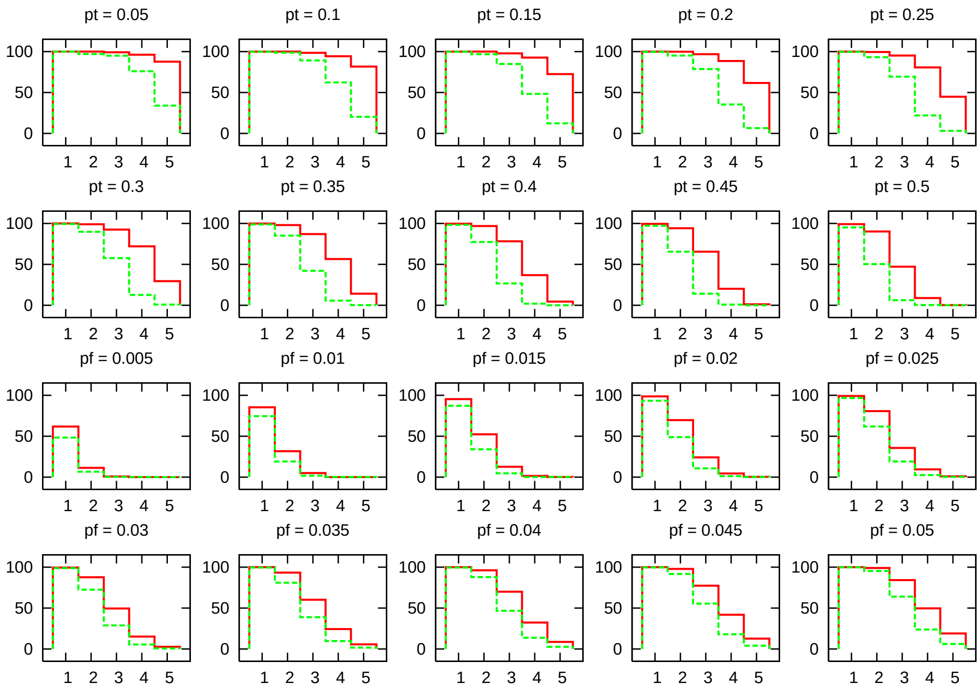

Figure 4: A bar diagram showing the number of runs during which an alarm decision was made under true alarm conditions with malfunction probability p_t and false alarms with malfunction probability p_f. The red histograms are for $n = 200$ and the green dotted-line histograms for $n = 150$.

In Figure 5, we examine the "success rates" of the network in making the correct decision: the sum of true positives over the repetitions is given for each parameter set, as well as the sum of false positives. Assuming that the cost of a false positive is equal to that caused by a false negative, the difference of these sums can be used as a measure of success for a decision-making setup (i.e., the number of neighbors γ used to trigger an alarm). For each value of γ, we have added over all values of p_t and p_f used in the experiments. Note that also other costs for the false negatives and false positives could be used in analyzing the output of the simulator. It is evident that a sparse network is highly vulnerable to false positives as it is only able

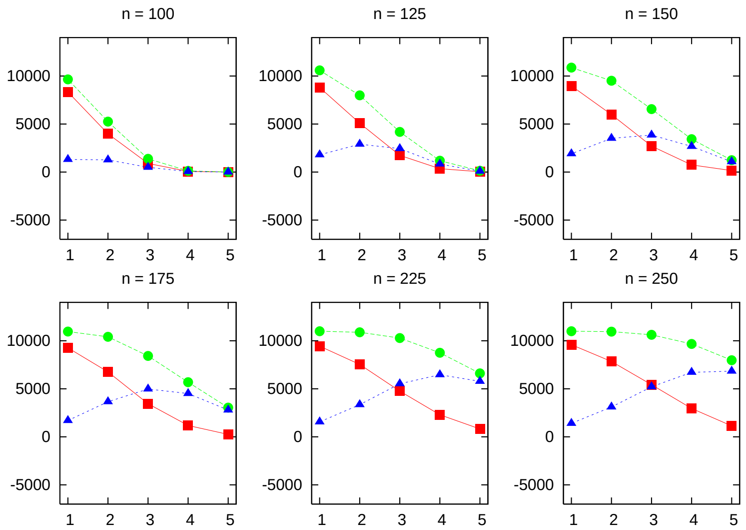

Figure 5: The sums of $(n, 0.2)$-runs that end in alert decision over the different p_t and p_f for comparing the different values of γ; in addition to the sums for the true (●) and false (■) alerts, the difference (▲)curves of the two sums are shown.

to properly propagate true positives when a low value of γ is used. Hence it seems more feasible to build a dense network and use a higher value for γ, as this optimizes the cost-efficiency in the sense of how many more true positive alarms there are than false positives.

5.2 Range variation experiments

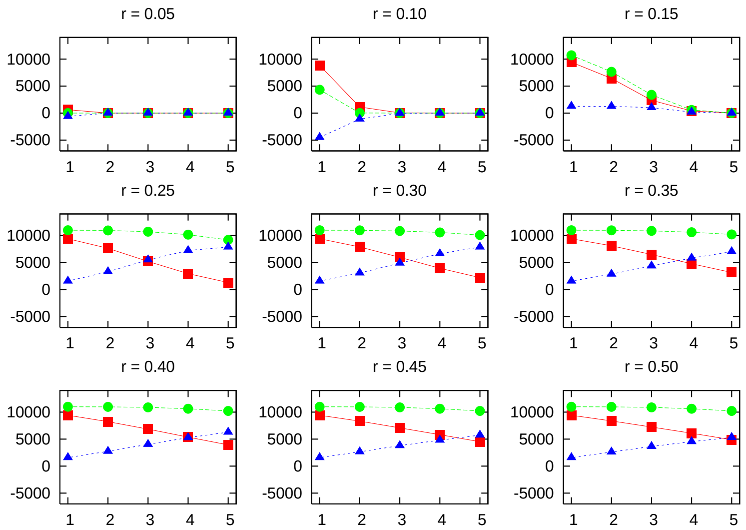

Figure 6: The sums of (200, r)-runs over the values of p_t and p_f for each γ; ■-plots are the false positives, ●-plots the true positives, and ▲-plots are the difference of the former two.

In Figure 6 it is seen that for a too small of a range r, no signal succeeds in propagating to the target area, neither true nor false alarms. For $n = 200$, good range values are 0.25 and 0.30; for higher values of r, the propagation of false positives quickly accelerates, which reduces the overall reliability of the network.

5.3 Alert and target area variation experiments

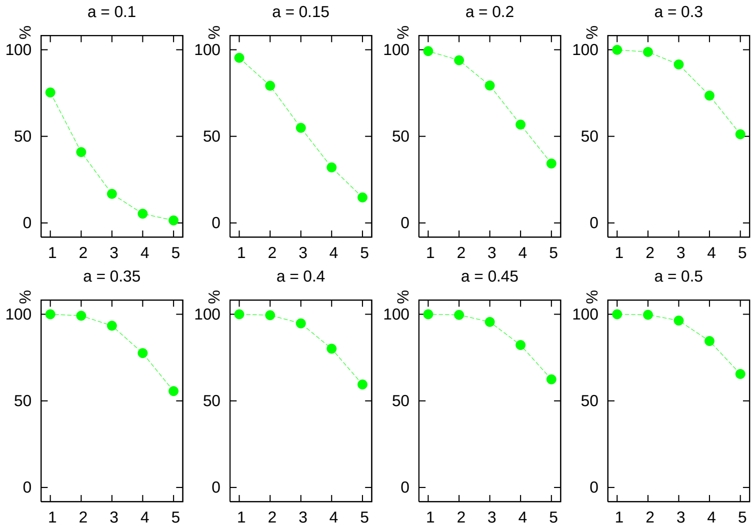

Figure 7: The sums over values of p_t and p_f for each γ for the true positives for the alert area variation runs, with $a = A_\ell(x) = A_\ell(y)$ and $A_u(x) = A_u(y) = 0$.

Figure 7 shows that when a low value of γ is used, the propagation of a true alarm to the target area is rather unaffected by the size of the alert area, but for higher values of γ, a larger alert area is, as intuitive, more likely to cause an alarm in the target area. No false-positive tests were ran, as when an alert area is actually present, each alarm that makes it to the target area is in fact a useful alarm, regardless of how it originated.

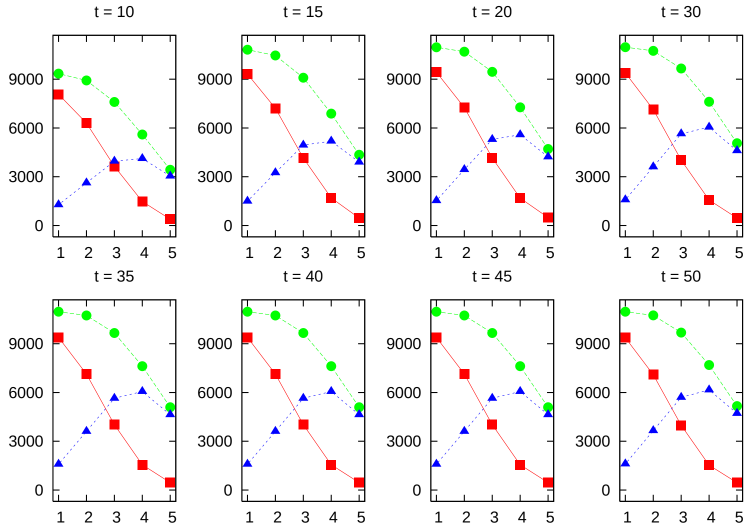

Figure 8: The sums over values of p_t and p_f for each γ for the true positives for the target area variation runs, with $t = T_u(x) = T_u(y)$ and $T_\ell(x) = T_\ell(y) = 1$.

For the target area alteration (plotted in Figure 8, we found that $\gamma = 4$ remains the best choice for the (200, 0.2) setting. Increasing the target area is in practice costly, as the model allows no failures within the target area and hence that area would need to be supervised with careful attention. All in all, the effect of

Table 3: A recommendation on the number of neighbors γ_{opt} to use in decision-triggering for different network setups (n, r).

n	r	γ_{opt}	n	r	γ_{opt}
100	0.2	1	200	0.15	2
125	0.2	2	200	0.25	5
150	0.2	3	200	0.3	5
175	0.2	3	200	0.35	5
200	0.2	4	200	0.4	5
225	0.2	4	200	0.45	5
250	0.2	5	200	0.5	5

the target area variation is not very significant — the only noticeable jump in performance occurs between $t = 0.1$ and $t = 0.15$.

6 Applications

A sensor network can be deployed to monitor both hostile and benign phenomena, such as toxicity levels or the functioning of an assembly line [8]. Also ecologists are beginning to employ sensor networks in *habitat monitoring* to study e.g. population dynamics [10]. Observing and forecasting natural disasters such as earthquakes, tornados and forest fires is yet another possible application [5, 11]. Also traffic control applications exist: monitoring traffic jams by placing sensors in taxi cabs or calculating the number of available parking spaces [11]. We can also foresee military and security applications, such as replacing land mines that pose a difficulty when disarming is required with a sensor network that detects intrusion and alerts an external defense system instead of reacting itself — this would also provide a strategic benefit, as the intruder would not be immediately aware of having been noticed. In general terms, simulation studies such as these are most useful in optimizing the composition of a sensor network when the costs of false positives and negatives can be estimated and sensors and their count can be tuned to fit the desired behavior.

7 Conclusions

In this study we have examined the effects of design variables for a sensor network in choosing an appropriate decision-making criterion for propagating an alert in a network where false positives are possible and undesirable. Increasing the density of the network — either by adjusting the number of sensor per unit square n or their communication range r — naturally increases the number of neighbors that each sensor has. Increasing the decision threshold γ, i.e., the number of alerting neighbors that causes a sensor to output an alert itself, monotonously as the number of neighbors reduces the risk of false positives, but also decreases the success of propagation for true positives.

Assigning a cost to a false positive and a gain for a true positive (or equivalently a cost to a false negative), one may choose a value of γ that optimizes the total cost. We have used an unjustified but application-neutral assumption in our analysis that the cost of a lost alert and the cost of a false alarm were equal; in many cases the cost of a lost alert can be significantly higher. It is however straightforward to analyze the simulator output also for other cost assignments.

If the expected area to be covered by a true alert is small, the decision threshold must be low with respect to the number of neighbors (this could occur for example with a forest fire, that initiates on a small area), but for large-area alerts (such as minor earthquakes), a sparser network with a higher γ will be usable.

Increasing the target area is only wise if the cost of ensuring proper operation within the area is not costly, as an increase in the target area does not drastically affect the decision-making behavior.

As further work, it would be of interest to study mathematically the relation between network density and the best decision threshold, as well as to examine the behavior of the network under multiple failure modes. Giving each sensor an exponentially distributed lifetime after which it fails and a distribution over the different failure modes, a more realistic model would be achieved, however making the analysis more complex as the number of degrees of freedom would increase. For a realistic model with numerous parameters, we would like to see whether using a genetic algorithm to tune them optimally (cf. the cellular automata in [4].

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Maganize*, pages 102–114, Aug. 2002.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38:393–422, 2002.

[3] C.-F. Huang and Y.-C. Tseng. The coverage problem in a wireless sensor network. In *Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications*, New York, NY, USA, Sept. 2003. ACM Press.

[4] M. Mitchell, J. P. Crutchfield, and P. T. Hraber. Evolving cellular automata to perform computations: Mechanisms and impediments. *Physica D*, 75:361–391, 1994.

[5] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58, May 2000.

[6] G. Qiang, K. J. Blow, D. J. Holding, and I. Marshall. Determining design parameters for ad hoc wireless sensor networks. In *Proceecings of EFTA'03: Emerging Technologies and Factory Automation*, volume 1, pages 545–550. IEEE, 2003.

[7] C. S. Raghavendra, K. M. Sivalingam, and T. Znati, editors. *Wireless Sensor Networks*. Kluwer Academic Publishers, Boston, MA, USA, 2004.

[8] A. Sinha and A. Chandrakasan. Dynamic power management in wireless sensor networks. *IEEE Design & Test of Computers*, 18(2):62–74, Mar./Apr. 2001.

[9] Social Science Research Computing. RePast — software framework for creating agent based simulations using the Java language, 2004. Available online at `http://repast.sourceforge.net/`.

[10] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin. Wireless sensor networks: Habitat monitoring with sensor networks. *Communications of the ACM*, 47(6), June 2004.

[11] S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman. A taxonomy of wireless micro-sensor network models. *ACM SIGMOBILE Mobile Computing and Communications Review*, 6(2), Apr. 2002.

[12] H. Zhang and J. Hou. Energy efficiency: On deriving the upper bound of α-lifetime for large sensor networks. In *Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, New York, NY, USA, May 2004. ACM Press.

Acknowledgments

The first author is currently supported by the Academy of Finland under grant 206235 and the Nokia Foundation. We thank Intel Corporation for running the RePast experiments. The participants and faculty of the SFI CSSS 2004 as well as Professor Pekka Orponen of Helsinki University of Technology we thank for their valuable comments.