

Distributed Algorithm for a Mobile Wireless Sensor Network for Optimal Coverage of Non-stationary Signals

Andrea Kulakov
Computer Science Department
Faculty of Electrical Engineering
Skopje, Macedonia

kulak@etf.ukim.edu.mk

Danco Davcev
Computer Science Department
Faculty of Electrical Engineering
Skopje, Macedonia

etfdav@etf.ukim.edu.mk

ABSTRACT

In this paper we will deal with the problem of optimal coverage of a wireless sensor network for random signals appearing with non-stationary distributions. The wireless sensor network can be either with limited mobility or with large redundancy of the nodes. We will give a distributed algorithm that successfully solves this problem, and we will show its efficiency in simulations in a 2-D environment.

INTRODUCTION

A sensor network consists of a large number of sensors distributed randomly in a geographical region. In order to reduce sensing uncertainty which arises from limitations associated with *physical* configuration of sensor network nodes, *physical reconfiguration* in the form of a coordinated mobility of the nodes is proposed in so called Networked Infomechanical Systems [1]. A combination of both event detection and an awareness of sensing uncertainty level produce a trigger for reactive coordinated mobility of mobile sensors and redeployment of nodes. The nodes can reconfigure either in one, two or three dimensions. In this paper we will demonstrate results obtained with simulations in 2-D environments although they can be extended to 1-D and 3-D environments as well.

The problem for optimal coverage, on the other hand, can be also treated in wireless networks with large redundancy of the nodes. Since many of

the sensor readings on near-by sensors will be highly correlated (if not the same), some of them can be chosen as active and the rest can be put in a sleep mode in order to save energy for future use.

We introduce a distributed algorithm for optimal coverage of non-stationary discrete signals which is based on Fritzke's Growing Neural Gas model of an artificial neural network [2]. We have also improved the algorithm in a way that it converges faster to the optimal coverage when the signal distribution suddenly changes.

Also we have treated the problem of network connectivity and have changed the original algorithm such that it does not allow the network to be separated in two distinct sub-networks.

We assume that each sensor is capable of sensing a well-defined *convex* region S around itself called the *sensing region*. Each sensor also has a radio interface and can communicate directly with some of the sensors around it. In the simulations we assume that the sensors know its position in the field (either by GPS or other GPS-less localization method, e.g. [3], which is not treated in this paper) and can move restrictedly in each direction on a 2-D plane.

THE ALGORITHM

We will first give the algorithm in slightly modified version of the original algorithm proposed by Fritzke for artificial neural networks which was supposed to work on a single computer. This version is running on each of the sensor nodes. All the sensor units have some random position w_i in \mathbf{R}^2 .

1. Listen for the input signal ξ , distributed according to some distribution $P(\xi)$.
2. Find the nearest sensor unit s_1 and the second-nearest unit s_2 among several nodes that detected the signal.
3. Increment the age of all edges (radio connections) emanating from s_1 .
4. Add the squared distance between the input signal and the nearest unit in input space to a local error variable:

$$\Delta error(s_1) = \|w_{s_1} - \xi\|^2$$

5. Move s_1 and its direct topological neighbors toward ξ by fractions ε_b and ε_n , respectively, of the total distance:

$$\Delta w_{s_1} = \varepsilon_b (\xi - w_{s_1})$$

$$\Delta w_n = \varepsilon_n (\xi - w_n) \text{ for all neighbors } n \text{ of } s_1$$

6. If s_1 and s_2 are connected by an edge, set the age of this edge to zero. If such an edge does not exist, create it.
7. Remove edges with an age larger than a_{max} . If this results in sensor nodes having no emanating edges, set these nodes as “free”.
8. If the number of input signals generated so far is an integer multiple of a parameter λ , activate some “free” unit (node) as follows:
 - Determine the unit q with the maximum accumulated error.
 - Broadcast an “order” to insert a new unit r halfway between q and its neighbor f with the largest error variable: $w_r = 0.5(w_q + w_f)$.
 - The nearest “free” unit should accept the order and try to move towards the requested position.
 - Insert edges connecting the new sensor unit r with the units q and f , and remove the original edge between q and f .
 - Decrease the error variables of q and f by manipulating them with a constant α . Initialize the error variable of r with the new value of the error variable of q .
9. Decrease all local error variables by multiplying them with a constant d .
10. If a stopping criterion (e.g., network size or some performance measure like minimal overall error) is not yet fulfilled, go to step 1.

It is obvious that step 2, for example, takes more than one processing step to be determined.

We have further improved this algorithm by means of manipulating the parameter ε_n which determines the ratio by which the neighboring sensor nodes will be affected to move towards the sensor node which sensed the input signal most intensely.

Namely when the signal distribution suddenly changes, then the overall error which is a sum of all local error variables, will enormously increase. This will trigger increase of the parameter ε_n which will result in more vivid mobility of the sensor nodes toward the new distribution of the signals. After a while when the moving average of the global error becomes relatively constant, that means that no further improvement is possible with this value of the parameter ε_n . Then the parameter is switched back to its low value which causes less mobility of the sensor nodes and allows better spreading of the nodes within the region of the current distribution of the signals.

SIMULATION RESULTS

We have conducted the experiments with the simulation testbed provided courtesy of Hartmut S. Loos and Bernd Fritzke at <ftp://ftp.neuroinformatik.ruhr-uni-bochum.de/pub/software/NN/DemoGNG/DemoGNG-1.5.zip> on which we have added the control of the parameter ε_n as explained before.

We have compared our improvement of the algorithm with the proposed solution by Fritzke in [4] where he introduces a measure of utility of each node, with possibility of eliminating nodes which are not used for a long period. This solution is not acceptable for implementation in sensor systems because we can not afford to eliminate certain nodes, but rather we redistribute them more agile according to the new signal distribution.

In Figure 1 the nodes of the sensor network are divided into two tightly connected sub-networks of sensor nodes. Small dots in the subsequent figures show the appearance of the last 200 input signals. The areas shown with lighter color in figures show the probability distributions in the 2-D plane, where the concentration of the input signals is uniform. In Figure 1 we can see a sudden change of the probability distribution from two rectangular areas where the sensor nodes were previously distributed, into one ring.

In Figures 2-5 it can be seen how the sensor networks move toward the areas with biggest concentration of signals when the distribution changes from rectangular to ring. Once the network gets adapted to this distribution we suddenly change the distribution in our simulations back to rectangular. In Figures 6-10 it can be seen the behavior of the network nodes while adapting to the new distribution.

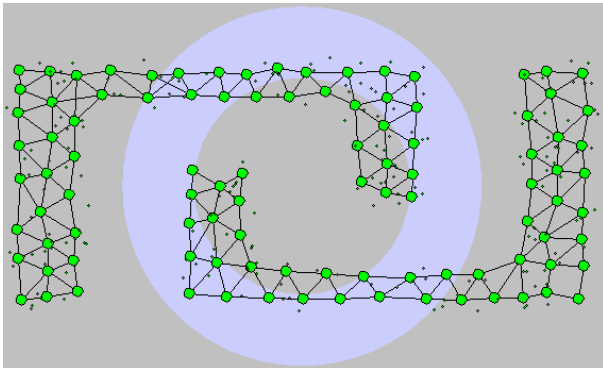


Figure 1 Sudden change of the distribution of the signals results in poorly covered signals with sensor nodes

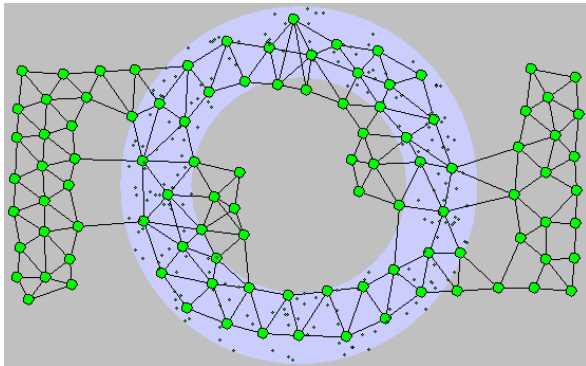


Figure 2 Sensor nodes slowly begin to move toward the biggest concentration of input signals

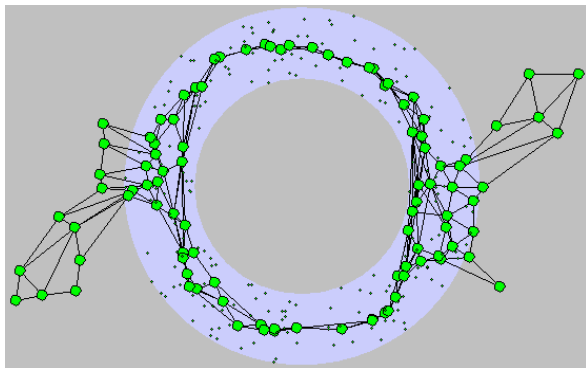


Figure 3 With the increase of the parameter $\varepsilon_n=0.2$, the nodes start swiftly to move towards the area of biggest concentration of input signals

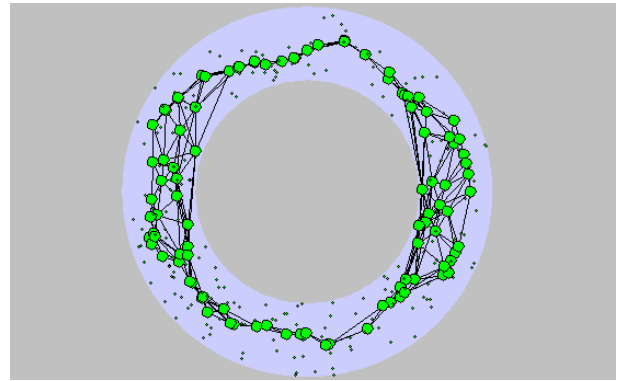


Figure 4 With large ε_n parameter, all the nodes soon become congregated inside the area of biggest concentration of input signals

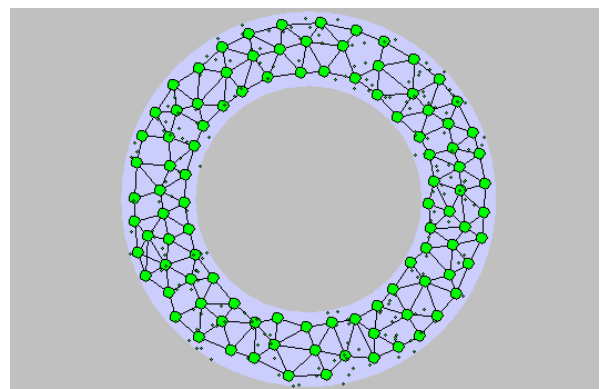


Figure 5 Subsequent low parameter $\varepsilon_n=0.0001$, leads to better dispersion of sensor nodes inside the area of biggest concentration of input signals and thus to lower overall error

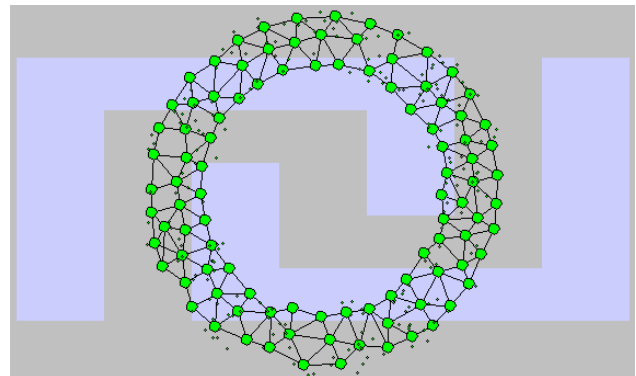


Figure 6 Sudden change of the probability distribution from ring back to two rectangular areas

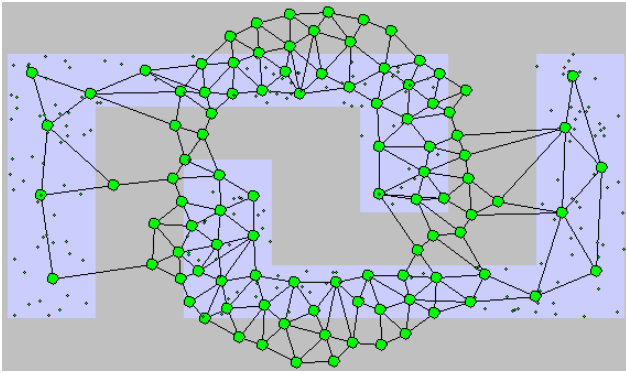


Figure 7 Sensor nodes begin slowly to cover the new distribution of the input signals

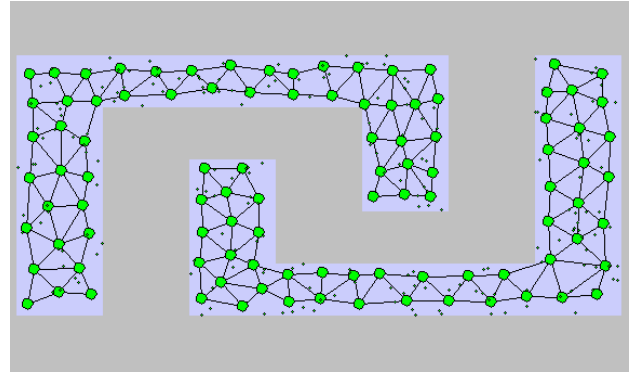


Figure 10 Later on, low ϵ_n parameter, leads to better dispersion of nodes and thus to lower overall error

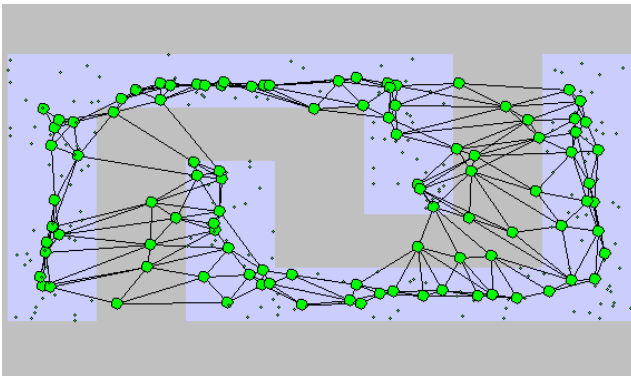


Figure 8 With the increase of the parameter ϵ_n , the nodes start again to swiftly move towards the area of biggest concentration of input signals

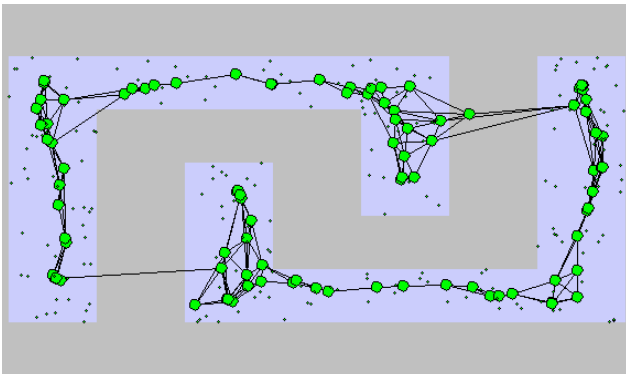


Figure 9 With large ϵ_n parameter, most of the nodes soon become distributed inside the area of biggest concentration of input signals

When we compare the overall error from signals obtained with our algorithm with overall error from the GNG-Utility algorithm of Fritzke we show that the times to converge to optimal distribution of sensor nodes depending on the probability distribution of the input signals, are many times shorter in case of our algorithm. This can be seen from the Figures 11 and 12. At time interval around 1100 the probability distribution is changed from rectangular to ring, and then at time interval around 2100 it is switched back to rectangular.

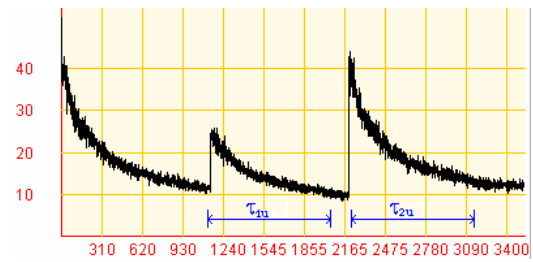


Figure 11 Times from onset of change of the probability distribution to the constant error level, τ_{1u} from rectangular to ring distribution, τ_{2u} from ring to rectangular distribution in case of a GNG-U algorithm

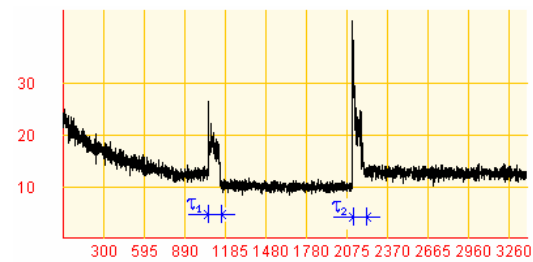


Figure 12 The same times as in Figure 11, from onset of change of the probability distribution to the constant error level, in case of our algorithm

Another advantage of our algorithm over GNG-Utility algorithm is that the starting overall errors in our case are usually much smaller since the sensor nodes are not disappearing from the network and then appearing again at a new location, but rather they are swiftly moving towards the new distribution of the input signals and constantly contributing to the better coverage of the signals with the sensor network.

We have used an algorithm from [5] where several problems from graph theory are solved, one of them being the connectivity problem. That algorithm can be modified in a distributed fashion and can be used to detect bridges between two sub-networks (see Figures 13 and 14).

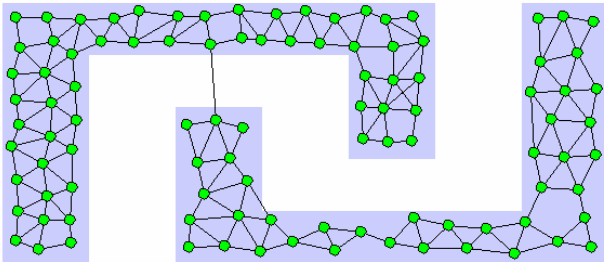


Figure 13 The same distribution of the nodes as in Figure 10, but this time the connectivity between the two parts is guaranteed.

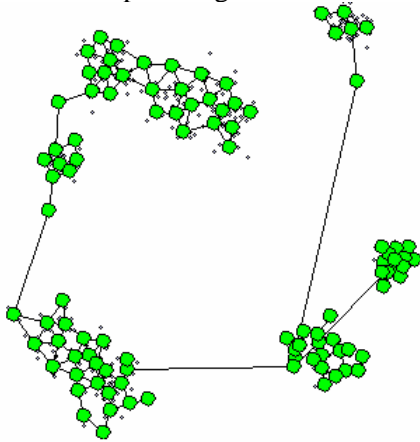


Figure 14 Connected network after being deployed over discrete distribution of signals

For future work we plan to constrain the distance between two neighboring nodes and to have a possibility to increase the number of bridges that connect different otherwise disjoint sub-networks.

REFERENCES

- [1] <http://deerhound.ats.ucla.edu> (NIMS Coordinated Mobility)
- [2] Fritzke, B., A Growing Neural Gas Network Learns Topologies, in Tesauro, G., Touretzky, D.S., and Leen, T.K. (eds.) *Advances in Neural Information Processing Systems 7*, MIT Press, Cambridge MA, USA, 1995.
- [3] Bulusu, N., Heidemann, J. and Estrin, D., "GPS-less low-cost outdoor localization for very small devices," *IEEE Personal Communication*, vol. 7, pp. 28–34, October 2000.
- [4] Fritzke, B., A self-organizing network that can follow non-stationary distributions. In *ICANN'97: International Conference on Artificial Neural Networks*, pages 613-618. Springer.
- [5] Sedgewick R., *Algorithms in Java*, Third Edition, Addison Wesley, 2003