# Optimal Wireless Sensor Network Layout with Metaheuristics: Solving a Large Scale Instance

Enrique Alba, Guillermo Molina

Departamento de Lenguajes y Ciencias de la Computación
University of Málaga, 29071 Málaga, Spain
eat@lcc.uma.es guillermo@lcc.uma.es

## 1 Introduction

Nowadays, the trend in telecommunication networks is having highly decentralized, multinode networks. From small, geographically close, size-limited local area networks the evolution has led to the huge worldwide Internet. This same path is being followed by wireless communications, where we can already see wireless telephony reaching virtually any city in the world.

Wireless networks started as being composed by a small number of devices connected to a central node. Recent technological developments have enabled smaller devices with computing capabilities to communicate in the absence of any infrastructure by forming ad-hoc networks. The next step in wireless communications begins with ad-hoc networks and goes towards a new paradigm: Wireless Sensor Networks (WSN) [1].

A WSN allows an administrator to automatically and remotely monitor almost any phenomenon with a precision unseen to the date. The use of multiple small cooperative devices yields a brand new horizon of possibilities yet offers a great amount of new problems to be solved.

We discuss in this paper an optimization problem existing in WSN: the layout (or coverage) problem [2, 3]. This problem consists in placing sensors so as to get the best possible coverage while saving as many sensors as possible. A genetic algorithm has already been used to solve an instance of this problem in [3]. In this paper we define a new instance for this problem, and tackle it using some metaheuristic techniques [4, 5, 6] and solve a large dimension instance.

This work is structured as follows. After this introduction, the WSN layout problem (WSN problem for short) will be presented, and its formulation described in Section 2. Section 3 explains the optimization techniques employed for solving this problem. Then in Section 4 the experiments performed and the results obtained are analyzed. Finally, Section 5 shows the conclusions and future work.

## 2 WSN Problem

In this section we describe the layout problem for WSN, then present the formulation employed for its resolution.

### 2.1 Problem Description

A Wireless Sensor Network allows to monitor some physical set of parameters in a region known as the *sensor field*. When a WSN is placed in the sensor field, every sensor monitors a region of the field; ideally the complete network is able to monitor all the field by adding all the pieces of information together. It is the duty of the designer to establish what is the sensor field that the WSN has to monitor.

A node sensing area (the area that a single sensor can sense) can be modelled with a circle whose radius $R_{SENS}$ -or *sensing radius*- indicates the sensing range of the sensor. The value of this range is determined by both the magnitude that is sensed and the sensor itself (hardware employed). Similarly, $R_{COMM}$, the *communication radius* of a sensor, defines the circle where any other sensor can establish a direct communication link with it. The value of this range depends on the environment, the radio hardware, the power employed and other factors.

When a WSN is deployed in the sensor field, the sensors form a wireless ad-hoc network in order to communicate their sensing results to a special station called the *High Energy Communication Node* (HECN). The data can then be analyzed by the HECN processor, or be accessed by the network administrator. Any sensor unable to transmit its sensing data to the HECN is useless. The sensing information is not sent through a direct link to the HECN, but rather a hop by hop communication is employed. Thus, for any node to be useful, it has to be within communication range of another useful node.

The sensing area of the WSN is the union of the individual sensing areas of all the useful nodes. The designer wants the network to cover the complete sensing area, or, if this is unfeasible, to cover as much of it as possible. On the other hand, the number of sensor nodes must be kept as low as possible, since using many nodes represents a high cost of the network, possibly influences the environment, and also provokes a high probability of detection (when stealth monitoring is desired).

The problem of designing the layout for a WSN can be defined as an extension of an existing problem: the radio network design problem (RND) [7]. The objective of this problem is to maximize the sensing area of the network while minimizing the number of sensors deployed.

### 2.2 Problem Formulation

For this work we employ a square terrain as the sensor field, and use a discrete model to represent it. This model is a $287 \times 287$ point grid as in [7], where every point can be either monitored or not.

Sensor nodes can only be placed in some of those field points. If a sensor can communicate with the HECN, then a discretized circular area around its location is considered to be monitored. The available field points for placing the sensors are given as an ordered list (the *Available Location Sites*, ALS for short) that constitutes the specific *problem instance*. Figure 3 shows a graphical example of a WSN instance (left) and a solution layout with its underlying topology (right).

```
t:= 0;
Initialize(T,Sa);
while not end_condition(t,Sa) do
        while not cooling_condition(t)
                Sn := Choose_neighbor(Sa);
                Evaluate(Sa,Sn);
                if Accept(Sa,Sn,T) then
                        Sa := Sn;
                end if
                t := t+1;
        end while
        Cooldown(T);
end while
```

**Fig. 1.** Pseudocode for SA

The WSN problem can be reduced to selecting from the list of available points a subset of locations that form the optimal sensor network. The list is ordered so that any bit string of the same length a the ALS represents a solution attempt to the problem (the '1's in the string indicating the chosen locations).

From the previous definition of the problem, a fitness function that combines both objectives is employed [7] (Equation 1). The objective is to **maximize** the fitness value of the solution.

$$f(\boldsymbol{x}) = \frac{Coverage(\boldsymbol{x})^2}{Nb.\ of\ sensors(\boldsymbol{x})}, \qquad Coverage(\boldsymbol{x}) = 100 \cdot \frac{Covered\ points}{Total\ points} \tag{1}$$

## 3 Optimization Techniques

In this section, we describe the two techniques used to solve the problem: simulated annealing and CHC.

### 3.1 SA Algorithm

Simulated annealing is a trajectory based optimization technique. It was first proposed by Kirkpatrick et al. in [5]. SA is a fairly commonly used algorithm that provides good results and constitutes an interesting method for comparing results and test other optimizing methods. The pseudocode for this algorithm is shown in Fig. 1.

The algorithm works iteratively and keeps a single tentative solution $S_a$ at any time. In every iteration, a new solution $S_n$ is generated from the old one, $S_a$, and depending on some acceptance criterion, it might replace it.

The acceptance criterion is the true core of the algorithm. It works as follows: both the old ($S_a$) and the new ($S_n$) solutions have an associated quality value - determined with a *fitness* function. If the new solution is better than the old one, then it will replace it. If it is worse there is still some chance that it will replace it. The replacing probability is calculated using the quality difference between both solutions and a special control parameter $T$ named *temperature*.

The acceptance criterion ensures a way of escaping local optima by choosing solutions that are actually worse than the previous one with some probability. That probability is calculated using Boltzmann's distribution function:

$$P = \frac{2}{1 + e^{\frac{fitness(S_a) - fitness(S_n)}{T}}} \tag{2}$$

As iterations go on, the value of the temperature parameter is progressively reduced following a cooling schedule, thus reducing the probability of choosing worse solutions and increasing the biasing of SA towards good solutions. In this work we employ a geometric rule, such that every $k$ (*Markov chain length*) iterations the temperature is updated as $T(n+1) = \alpha \cdot T(n)$, where $0 < \alpha < 1$ is called the temperature decay.

### 3.2 CHC Algorithm

The second algorithm we propose for solving the RND problem is Eshelman's CHC (*Cross generational elitist selection, Heterogenous recombination, and Cataclysmic mutation*), a kind of Evolutionary Algorithm (EA) surprisingly not used in many studies despite it has unique operations usually leading to very efficient and accurate results [6]. Like all EAs, it works with a set of solutions (*population*) at any time. The algorithm proceeds iteratively, producing new solutions at each iteration, some of which will be placed into the population replacing others that were previously included. The pseudocode for this algorithm is shown in Fig. 2.

The algorithm CHC works with a population of individuals (solutions) that we will refer to as $P_a$. In every step, a new set of solutions is produced by selecting pairs of solutions from the population (the parents) and recombining them. This selection is made in such a way that individuals that are too similar can not mate each other, and recombination is made using a special procedure known as HUX (*Half Uniform crossover*). This procedure copies first the common information for both parents into both offspring, then it translates half the diverging information from each parent to each of the offspring. This is done in order to preserve the maximum amount of diversity in the population, as no new diversity is introduced during the iteration (there is no mutation operator). The next population is formed by selecting the best individuals among the old population and the new set of solutions (elitist criterion).

As a result of this, at some point of the execution, population convergence is achieved, so the normal behavior of the algorithm should be to stall on it. A special mechanism is used to generate new diversity when this happens: the *restart* mechanism. When restarting, all of the solutions except the very best ones are significantly modified. This way, the best results of the previous phase of evolution are maintained and the algorithm can proceed again.

## 4 Tests and Results

In this section we describe the experiments and present the results obtained using the two algorithms described in Section 3. The results are then analyzed

```
t:=0;
Initialize(Pa,convergence_count);
while not ending_condition(t,Pa) do
        Parents := Selection_parents(Pa);
        Offspring := HUX(Parents);
        Evaluate(Offspring);
        Pn := Elitist_selection(Offspring,Pa);
        if not modified(Pa,Pn) then
                convergence_count := convergence_count-1;
                if (convergence_count == 0) then
                        Pn := Restart(Pa);
                        Initialize(convergence_count);
                end if
        end if
        t := t+1;
        Pa := Pn;
end while
```

**Fig. 2.** Pseudocode for CHC

rigorously in order to determine the statistical confidence of the observed differences.

The instance solved in this work is a very large instance (1000 available locations), specially if compared with the previously existing work [7] where the biggest instance had only 349 available locations. The sensor field is modelled by a $287 \times 287$ point grid. All sensors behave equally and both their sensing and communication radii are set to 22 terrain points. The ALS is formed by 1000 locations randomly distributed over the sensor field following a uniform distribution. Figure 3 illustrates the instance of the problem, and shows a random solution for this instance using 167 sensors and covering 56.76% of the sensor field. The low quality achieved by random search, the NP nature of the problem, and its high dimensionality clearly suggest the utilization of metaheuristics.

The models and parameters employed in our problem instance are summed up in Table 1.

| **Concept** | **Model** |
|---|---|
| Sensor Field | $287 \times 287$ point grid |
| ALS | 1000 points, uniform distribution |
| Solution | Bit string (1000 bits) |
| $R_{SENS}$ | 22 points |
| $R_{COMM}$ | 22 points |

**Table 1.** Models and parameters

The problem is solved using simulated annealing (SA) and CHC. The same instance of the problem is used for both algorithms, and a parameter tuning is made to get good results from them (the values of the parameters can be seen in Table 2). We will analyze the algorithm's effectiveness for solving the problem by inspecting the fitness obtained. The influence of the number of solution evaluations will also be studied by running several experiments with both algo-

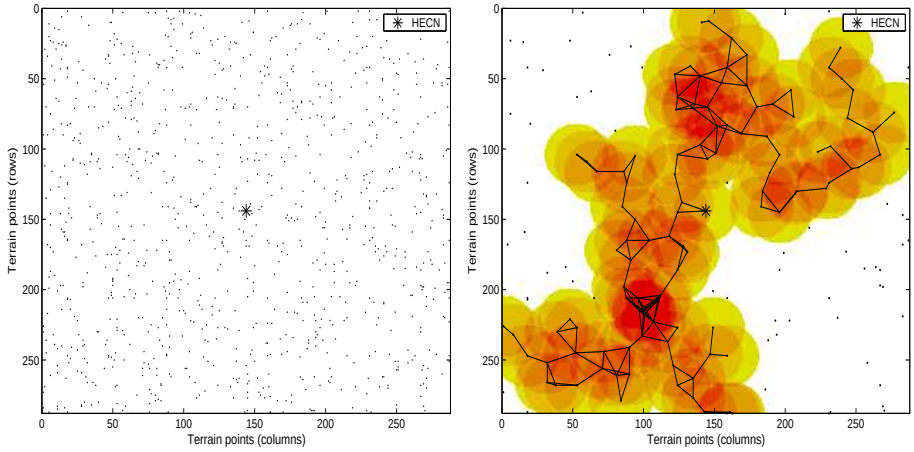


**Fig. 3.** Available sites in the problem instance (left), random solution (right)

rithms using increasingly higher number of allowed evaluations. The number of evaluations will range from $100,000$ up to $1,000,000$.

| | | | |
|---|---|---|---|
| Algorithm | CHC | Algorithm | SA |
| Population size | 100 | Mutation | *Bit flip prob. 1/Length* |
| Crossover | *HUX* | Markov-Chain length | 50 |
| Cataclysmic mutation | *Bit flip with prob.* 35% | Temperature decay | 0.99 |
| Incest threshold | 25% *of instance size* | Initial temperature | 1.05 |
| Selection of parents | *Random* | | |
| Selection of next generation | *Elitist* | | |

**Table 2.** Parameters of the algorithms

For every experiment the results are obtained by performing 30 independent runs, then averaging the fitness values obtained in order to ensure statistical confidence. Table 3 summarizes the results obtained for this study. Analysis of the data using Matlab's ANOVA/Kruskal-Wallis test plus Multcompare function has been used to get statistical confidence on the results with a confidence level of 95%. A minimum mean square error approximation function is calculated (from a list of standard functions) to estimate the relation between the average fitness value and the allowed number of evaluations, for both SA and CHC.

| **Evals.** | **50,000** | **100,000** | **200,000** | **300,000** | **400,000** | **500,000** | **1,000,000** |
|---|---|---|---|---|---|---|---|
| **SA** | 74.793 | 76.781 | 78.827 | 79.836 | 80.745 | 81.602 | 84.217 |
| **CHC** | 75.855 | 83.106 | 87.726 | 89.357 | 90.147 | 90.974 | **92.107** |

**Table 3.** Fitness results

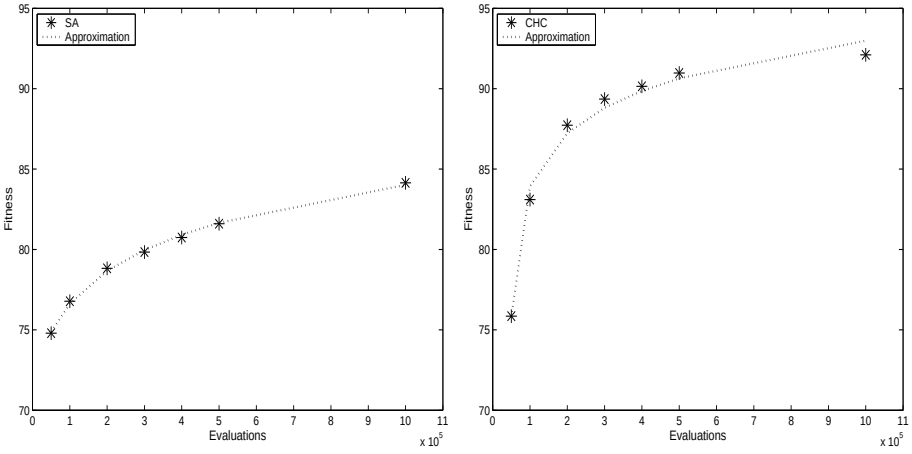


**Fig. 4.** Results obtained with SA and approximation (left), results obtained with CHC and approximation (right)

From the results in Table 3 we can state that the average fitness obtained with either SA or CHC improves when the number of evaluations is increased. In the first case (SA) the average fitness goes from 74.793 for $50,000$ evaluations to 84.217 for $1,000,000$ evaluations. In the second case (CHC) it goes from 75.855 to 92.107. Analysis of the data shows that the increment of the fitness values is meaningful for both algorithms when the difference in number of evaluations is bigger than $100,000$.

When it comes to comparing the two algorithms, CHC outperforms SA. The average fitness value obtained for any number of evaluations is greater using CHC than using SA. The analysis of the data confirms that CHC's results are significantly better than SA's for any number of evaluations except $50,000$, for which they are equivalent. Furthermore, the executions using CHC have outperformed the ones using SA that performed *five times* more solution evaluations. CHC with $100,000$ and $200,000$ evaluations has outperformed SA with $500,000$ and $1,000,000$ evaluations respectively (though analysis couldn't show the significance at 95% confidence). CHC with $200,000$ and $500,000$ evaluations is significantly better than SA with $500,000$ and $1,000,000$ evaluations respectively.

The improvement obtained augmenting the number of evaluations is sublineal and is best modelled for this range of values using a logarithmic function for both SA and CHC. Figure 4 shows the average fitness obtained by both algorithms in the different experiments as well as the mathematical models calculated for them. Equations 3 and 4 show the mathematical models for the fitness values obtained using SA and CHC respectively.

$$SA_{fitness}(evals) = 3.556 \cdot \log(evals/100,000 + 0.287) + 75.733 \quad (3)$$

$$CHC_{fitness}(evals) = 3.155 \cdot \log(evals/100,000 - 0.459) + 85.867 \quad (4)$$

## 5 Conclusions

We have defined a coverage problem for wireless sensor networks with its innate connectivity constraint. A very large instance containing $1,000$ available locations has been solved for this problem using two different metaheuristic techniques: simulated annealing and CHC.

CHC has been able to solve the problem more efficiently than SA. In our experiments CHC has been able to reach high fitness values with an effort (number of performed solution evaluations) less than five times smaller than the effort required by SA to reach that same fitness. The average fitness obtained by any of the algorithms improves if the allowed number of evaluations per execution is increased within the range employed for our experiments ($50,000$ to $1,000,000$ evaluations), however their growths are sublineal. Mathematical models for this dependence have been calculated for both algorithms, resulting in logarithmic functions modelling SA's and CHC's fitness growth.

In future work the effect of the relation between sensing and communication radii will be studied. We also plan to redefine the problem so as to be able to place the sensors anywhere in the sensor field (instead of only in the available positions), and also take into account the power constraints existing in WSN (much harder than in other systems).

## Acknowledgements

This paper has been partially funded by the Spanish Ministry of Education and Science and by European FEDER under contract TIN2005-08818-C04-01 (The OPLINK project, http://oplink.lcc.uma.es). Guillermo Molina is supported by grant AP2005-0914 from the Spanish government.

## References

[1] Akyildiz, I., Su, W., Sankasubramaniam, Y., Cayirci, E.: A survey on sensor networks. IEEE Communications Magazine (2002)
[2] Meguerdichian, S., Koushanfar, F., Potkonjak, M., Srivastava, M.B.: Coverage problems in wireless ad-hoc sensor networks. In: INFOCOM. (2001) 1380–1387
[3] Jourdan, D., de Weck, O.: Layout optimization for a wireless sensor network using a multi-objective genetic algorithm. In: Proceedings of the IEEE Semiannual Vehicular Technology Conference. Volume 5. (2004) 2466–2470
[4] Michalewicz, Z., Fogel, D.: How to Solve It: Modern Heuristics. Springer Verlag, Berlin Heidelberg (1998)
[5] Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **4598**(220) (1983) 671–680
[6] Eshelman, L.J.: The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination. In: Foundations of Genetic Algorithms, Morgan Kaufmann (1991) 265–283
[7] Alba, E., Molina, G., Chicano, F.: Optimal placement of antennae using metaheuristics. In: Numerical Methods and Applications (NM&A-2006), Borovets, Bulgaria (2006)