Placing Multiple Sinks in Time-Sensitive Wireless Sensor Networks using a Genetic Algorithm

Wint Yi Poe* and Jens B. Schmitt

disco — Distributed Computer Systems Lab, University of Kaiserslautern, Germany

Abstract. Performance issues in Wireless Sensor Networks (WSNs) play a vital role in many applications. Often the maximum allowable message transfer delay must be bounded in order to enable time-sensitive applications of WSNs like fire or intrusion detection systems. Hence, it is crucial to develop algorithms that minimize the worst-case delay in WSNs. In this work, we focus on the problem of placing multiple sinks such that the maximum worst-case delay is minimized while keeping the energy consumption as low as possible. For that purpose we develop an algorithm based on the Genetic Algorithm (GA) paradigm. To model and consequently control the worst-case delay of a given WSN we build upon the so-called sensor network calculus (a recent methodology introduced in [10]). In order to be able to assess the performance of the GA-based sink placement strategy we compare it to an exhaustive search algorithm as well as a Monte-Carlo search. All of the strategies are based on a discretization of the originally continuous search space into a finite search space, which forms a contribution of our work of independent interest. In the performance comparison with the other strategies the GA exhibits a favourable behaviour with respect to the quality of the solutions found and the computational effort invested. It thus seems to be a good candidate for addressing the problem of placing multiple sinks in large-scale time-sensitive WSNs.

Keywords. Wireless Sensor Networks, Optimal Sink Placement, Genetic Algorithm, Network Calculus, Worst-Case Delay.

1 Introduction

1.1 Background and Motivation

A WSN comprises a collection of autonomous sensing nodes, each equipped with modest computing ability and memory, a wireless transceiver, power source, and physical sensors [5]. The potential applications of WSNs are vast and include for example, environmental monitoring, security and surveillance, precision agriculture, utility plant monitoring, health monitoring, battlefield scenarios, building

* This work has been generously supported by Gottlieb Daimler- und Karl Benz-Stiftung under grant number 02-13/06, 2007.

management, and disaster recovery. Though, most WSNs are still mainly deployed for research purposes due to a lack of predictability and controllability. To be successful in commercial environments, predictable and controllable building blocks as well as analytical frameworks for WSNs are essential. Therefore, there are a number of components and functions of a WSN that need first to be looked at separately, before combining them to a comprehensive framework for predictable and controllable WSNs.

In many WSN applications, it is desired to collect the information acquired by sensors for processing, archiving and other purposes. The station where that information is required is usually called a sink or base station. A sink normally has higher capacity as well as cost than usual sensor nodes. Sinks can be sensors themselves or devices such as PDAs or gateways to other larger networks [5]. For large-scale WSNs, a single-sink model is not scalable since message transfer delays as well as energy consumption of the sensor nodes become prohibitive, due to the fact that most of the nodes would be far away from the sink and thus many hops must be traversed before the sink is reached. As a result, response times become excessive and the lifetime of the WSN becomes very short. Therefore, it is sensible to deploy multiple sinks so that messages reach their destination with less hops and consequently response times are decreased and energy is saved.

Although energy is usually considered the most critical resource in WSNs, some applications depend heavily on performance characteristics such as message transfer delays or buffer requirements. In particular, message transfer delay is the most critical issue for time-sensitive WSN applications like fire or intrusion detection systems in order to ensure a timely actuation in case of a detection event.

1.2 Assumptions and Problem Statement

As discussed above, if sinks are placed in good locations, this can reduce traffic flow and energy consumption for sensor nodes. In particular, we focus on strategies to minimize the maximum worst-case delay, which is important for any timely actuation based on the information collected by a WSN. Hence, a worst-case analysis approach is considered. While average-case analysis is useful in some applications, for many WSN applications like production surveillance or fire detection, it must be ensured that messages indicating dangerous information are not lost (with high probability) and arrive at the control center with minimum delay. The worst-case analysis can be performed by utilizing a new methodology called sensor network calculus [11]. Sensor network calculus is based on network calculus [7], which is a framework for worst-case analysis allowing to calculate maximum message transfer delays, maximum buffer requirements at each sensor node, and lower bounds on duty cycles. Sensor network calculus customizes the conventional network calculus to the typical setting in WSNs.

In WSNs, the sensor nodes must connect to at least one of the sinks in order to send data from their surroundings or from their neighboring nodes. In the latter case, the node acts as a router. So, it is obvious that some nodes are transferring relatively large amounts of data, even if they are not sources of

heavy traffic. As a result, these nodes run out of battery faster and response times become excessive, in particular, if nodes and sinks are placed in improper position. So, sink location affects the network performance, thus the sinks should be located as good as possible. However, that brings up the question how these sinks can be placed optimally in order to achieve a minimum message transfer delay for time-sensitive applications.

In mathematical terms we can pose the problem as follows:

$$min. \max_{i \in \{1,\ldots,n\}} \{d_i\}$$

with

$$d_i = f\left(\tau | \boldsymbol{\alpha}, \boldsymbol{\beta}\right)$$

$$\tau = g(\boldsymbol{s} | \boldsymbol{p}, \mathcal{R})$$

$$\boldsymbol{p} = \left(p_i^{(x)}, p_i^{(y)}\right)_{i=1,\ldots,n}$$

$$\boldsymbol{s} = \left(s_j^{(x)}, s_j^{(y)}\right)_{j=1,\ldots,k}$$

$$p_i, s_i \in \mathcal{F}$$

Here, n denotes the number of sensor nodes and $\boldsymbol{p}$ is the vector of their locations in the sensor field $\mathcal{F}$, these locations are assumed to be given. The values d_i are the worst case delays for each sensor node i. By minimizing the maximum worst-case delay in the field, it is ensured that response times are balanced as far as possible. k is the number of sinks and the vector $\boldsymbol{s}$ contains their locations, these locations are the actual decision variables of the optimization problem. This is somewhat hidden by the fact, that the delays d_i are only indirectly affected by the choice of the sink locations. In fact, as a first-order effect, the d_i are a function of the topology τ, in which the WSN organized the data flow towards the sinks, and the arrival and service curves of the sensor nodes, denoted as $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ in the formulae above. While the arrival and service curves are parameters for a given WSN scenario, the topology is itself a function of the nodes' locations, the routing algorithm $\mathcal{R}$ and the sinks' locations $\boldsymbol{s}$, where however only the sinks' locations are variable and the other two are again given parameters. Note, in particular, that we assume the routing algorithm to be given and not to be subject to the optimization. Although this could in principle be done, it would aggravate the problem further and is therefore left for further study.

So, in principle, we face a continuous optimization problem where the objective function is to minimize the maximum worst case delay in the field subject to constraints that ensure that each sensor node is connected to a sink (possible via multi-hop communication, determined by the routing algorithm) as well as some geographic constraints. Due to the highly non-linear, jumpy behaviour of

the worst case delay function f, and thus of the objective function, which results from the formulas derived by sensor network calculus (see [10]), the direct solution of that optimization problem is practically infeasible.

Therefore, the goal of our work is to develop a heuristic strategy for the sink placement problem that minimizes the maximum worst-case delay in WSNs based on the sensor network calculus framework. To this end, we propose a Genetic Algorithm-based heuristic for sink placement (GASP) and analyze its performance in comparison to two other strategies. One is an optimal strategy based on an exhaustive search which serves as an upper bound for the performance achievable by the GASP, and the other one is a Monte-Carlo based strategy that should serve as lower bound on the performance that can be expected from the GA. Note that all strategies are designated for networks in which all the sensor nodes' locations are known.

1.3 Outline

In general, optimal sink placement is a hard problem like many other location problems (it exhibits particular similarity to the uncapacitated warehouse location problem, an NP-complete problem). First, we introduce a method to discretize the search space resulting in a set of candidate locations for the placement of sinks in a WSN. These candidate locations are based on the concept of regions of indifference, that is regions for which wherever a sink is placed the routing topology will not be altered and thus the objective function for the worst-case delay does not change. Hence any location in such a region of indifference can be chosen as a candidate location. Nevertheless, for large-scale WSNs the number of candidate locations still grows very fast and an exhaustive search becomes quickly prohibitive. That is why we consider a GA-based strategy, although, of course, it cannot guarantee to arrive at a provably optimal solution.

The organization of the paper is as follows: In the next section, we introduce the GASP algorithms alongside with the method for discretization of the search space. Next, we analyze the performance of the three mentioned strategies (GASP, optimal, Monte-Carlo) in Section 3. Section 4 presents related work and Section 5 concludes the paper.

2 Genetic Algorithm-Based Sink Placement in WSNs

In this section, we first propose a discretization method that reduces the continuous search space into a finite set of candidate locations. Each candidate location represents a so-called region of indifference, in which wherever we place a sink the routing topology is not changed and thus also the value of the objective function, the maximum worst-case delay, is not altered. So, interestingly, the discretization of the search space does not result in a loss of optimality. Yet, as we still face a hard combinatorial optimization problem similar to warehouse location problems (which are known to be NP-complete) where however the objective function is highly non-linear in contrast to those, the only resort for larger

problem instances is a heuristic search technique. In the following two subsections, we discuss the method for the discretization of the search space and the interior workings of the GA-based sink placement. A more detailed presentation can be found in [9].

2.1 Discretization of the Search Space

We assume the locations of the sensors and their transmission ranges to be given. In a first step, we use the nodes' locations to calculate the total number of regions of indifference, so that we know the cardinality of the candidate location set. To do so, we look at the intersection regions of sensor nodes' transmission ranges. In that context, nodes are called neighbours if their transmission areas intersect. By the sensor nodes' transmission areas we obtain a tesselation of the sensor field. The atomic regions forming that tesselation become the regions of indifference for the sink placement problem, because inside such a region it does not matter where we place a sink since the routing topology remains invariant and thus the objective function remains unchanged. Thus, within such a region of indifference we can choose any location as candidate location and consequently discretize the search space.

To the best of our knowledge, there has been no explicit formula for the number of intersecting regions when their location and transmission range are given. In Figure 1, we discuss possible circle intersections and the way to calculate the total number of regions as given by our formula which is given in Equation 1. In our method, first, we pick a node i with all intersecting neighbors and for each neighbor we add two regions to the total number of regions. Then we eliminate node i from the scenario. Based on this a loop invariant can be formulated which shows the correctness of this procedure. The same procedure will continue until the last node which has no further neighbors. Notice that the network is connected so that no further neighbors node means the last node of our scenario and its neighbors have already been counted. Finally, we add a region to the total number of regions for the last node.

The formula to calculate the total number of regions of indifference, $N(n)$, for multi-circle intersections is given in Equation (1) and we give evidence for it with some graphical illustration in Figure 1.

$$N(n) = \left(\sum_{i=1}^{n-1} 2 * nb_i \right) + 1 \tag{1}$$

where

$$n := \text{number of sensor nodes}$$

$$nb_i := \text{number of neighboring nodes after elimination of nodes } 1, \ldots, i-1$$

$N(n)$is important as it controls a sampling process of the sensor field, where we iteratively increase the sampling as much as necessary until the maximum

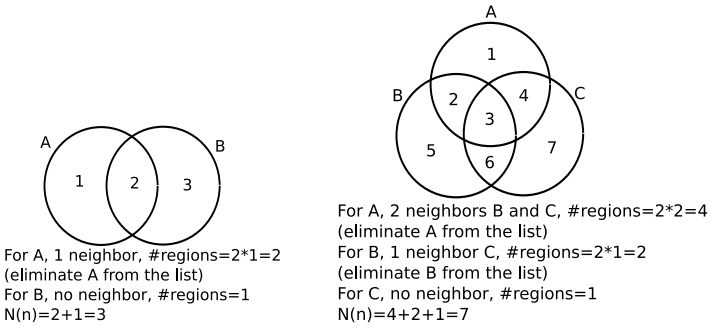

Fig. 1. Multi-circle intersection regions.

number of candidate locations is found. The sampling process works by laying a grid over the sensor field, determining for each grid point its neighborhood in terms of the transmission areas in which it is contained and then merging grid points which have identical neighborhoods. If the sampling accuracy is eventually high enough, all regions of indifference have been identified and the respective remaining grid points can be used as candidate locations for the sink placement. While the iterative grid-based procedure may seem computationally intensive and a direct method for calculating the indifference regions based on the locations and transmission ranges of the sensor nodes seems appealing, we had to find out after intensive literature research that the basic geometric problem has not been addressed so far and that even the easier problem of counting the intersection regions had not been addressed, to the best of our knowledge. However, the discretization never was a bottleneck in the practical scenarios we investigated and further has the advantage that it also works if the transmission properties cannot be captured by the typical circular structure of free-space signal loss models.

The alert reader might have noticed that there are some special cases of node constellations which do not agree with Equation (1) resulting in unwanted or duplicated regions. These special cases are shown in Figure 2. One such case is a cyclic dependency which results in an unwanted region that is useless for sink placement, because sensors cannot reach that region. Moreover, there is the possibility of scattered regions of indifference, where two or more regions of indifference result in the same routing topology so that actually one of them is sufficient. The discretization disposes with such areas in order to keep the search space minimal. The last case in the figure does not agree with Equation (1) because some regions of indifference might actually degenerate to a single point in the plane if we have a common intersection point between three or more transmission areas of sensor nodes. This case we rule out based on the observation that in practice this is very unlikely to happen.

So, the exact number of regions of indifference may differ from Equation (1) due to some cyclic dependencies and scattered indifference regions but is still under control of the discretization process. In the end, the search space becomes

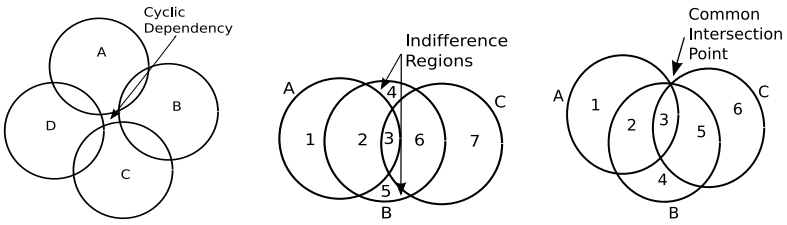

Fig. 2. (a) Cyclic Dependency (b) Indifference Regions (c) Common Intersection Point.

a finite set of candidate locations on which the following GA-based heuristic as well as the schemes we present in Section 3 can be built.

2.2 The Genetic Algorithm Based Sink Placement Strategy

Based on the set of candidate locations for the sinks an exhaustive search can be implemented that tries all possible combinations of sink placements. However, such an exhaustive search, while ensuring to find the optimal sink placement, becomes computationally too expensive for even moderately sized WSNs. More specifically, if the cardinality of the set of candidate locations is denoted by L and k sinks shall be placed, then the number of combinations to be checked becomes $\binom{L}{k}$, which grows quickly with L. So, the only resort from this is to use heuristic techniques. We decided to design a genetic algorithm based heuristic because the GA paradigm is known to be very flexible and allows to integrate domain-specific knowledge well [3].

Some Background on GAs John H. Holland [4] introduced the term and basic concepts of Genetic Algorithms in order to solve complex optimization problems. While being a random search technique, like a Monte-Carlo strategy, a GA attempts to learn as much as possible from its "experience" with the function to be optimized and can actually be reasoned to do so in an optimal fashion due to its inherent parallelism (see [3] for details). While not ensuring optimality for the solution found, it usually at least provides very good approximations as long as the so-called building block hypothesis is satisfied, i.e., if a good solution can be composed from good partial solutions (yet not in a very strict sense). The art of GAs therefore lies in the definition of the chromosomes of the individuals that represent the possible solutions for the problem at hand such that the building block hypothesis is fulfilled.

To implement a GA, the first step is to select initial individuals from the overall search space. Each individual represents a solution of the problem and is coded in the so-called chromosome. The selected set of individuals becomes the initial "population". The population size may vary depending on the search space and the problem characteristics. The next step is to generate new individuals by reproduction, which is often called the crossover operator of GAs, and by mutation. How to exactly implement these operators is problem-specific

[3], although a large set of variants to choose from exists. In the last step each individual's "fitness" is computed using the objective funtion, before the final operator of GA, the selection, is invoked to create the next "generation" of individuals which form the new population. Again, also for the selection operator there are many variants one can choose from. Which is best depends again on the problem and is usually subject to experimentation. The termination of a GA can be done in two fashions, either the population is converged, meaning that the individuals are all the same or at least very close to it, or, more commonly, the number of generations, i.e., the number of evolutionary loops is limited to some number. The latter criterion gives better control on the amount of computation that is invested.

Overview of the Genetic Algorithm-Based Sink Placement Strategy The GASP algorithm is given in Algorithm 1. Initially, it computes the candidate location set based on the method from Section 2.1. Each candidate location is given an index number where the indices are assigned according to an increasing distance towards the left upper corner of the sensor field, thus, to some degree, keeping track of the geographical positions of the candidate locations. [1]

Next, an initialization step is performed: select N individuals randomly by choosing a random subset of the candidate location set with the cardinality of the subset being equal to the number of sinks to be placed. These subsets are then transformed into an ordered list where the ordering criterion is according to the distance of the left upper corner of the sensor field, i.e., the chosen locations are ordered according to their respective index numbers. This chromosome encoding shall ensure that building blocks can actually be formed and that for the GA crossover operators (as described below) such building blocks are destroyed with low probability. Now, all individuals of the initial population are calculated for their fitness invoking the sensor network calculus computations to provide the maximum worst-case delay for the given sink placement.

Afterwards, the evolutionary loop that is composed of several GA operations can be started. Before actually performing the crossover operation, we need to pick two parents. The details of the problem-specific crossover operator we propose are discussed below, yet the idea of the crossover is to combine good partial solutions to achieve a better total solution. After the crossover, the mutation operator is invoked for each individual (old and new ones). Again the specifics of the mutation operator are deferred to the next subsection. The idea of the mutation operator is to integrate the explorative power of random search by trying new options in the search space which were not explored so far. Yet, it must be used with care, since otherwise the GA might degenerate into a pure random (Monte-Carlo) search. As last step in the evolutionary loop, the selection operator chooses the N best individuals from the whole set of old and new individuals to form the next generation (elitist strategy). The number of individuals per generation drives the degree of parallelism in the search and

[1] This is under the assumption of a rectangular field, which, however, does not pose a restriction as the sensor field can always be embedded into a rectangle.

Algorithm 1 Genetic Algorithm Sink Placement (GASP).

Given: location of the n sensor nodes, transmission ranges of the nodes, number of sinks to be placed (k)
1. Calculate the candidate locations set
2. Init: N individuals
   (i) Place the number of sinks ($s_1^{(i)}$,$s_2^{(i)}$,...,$s_k^{(i)}$) randomly from candidate locations set oriented from the left upper corner of the network field
   (ii) Calculate fitness function of maximum worst case delay
3. Evolutionary loop: G generations
   (i) Crossover
   (ii) Mutation
   (iii) Selection
4. go back to 3.

is again problem-specific, but in general the larger the search space, the larger should be the population size in order to avoid a premature convergence. After the selection of the next generation the evolutionary loop is repeated for the new population until a certain number of generations is reached. We choose to control the termination via an explicit number of generations in order to keep the amount of computation invested under control with, as discussed above. This also eases the comparison with other strategies. As will be discussed in the performance evaluation section, for larger problems we need to choose the number of generations larger in order to achieve good solutions.

The GASP Operators In this section, we discuss in more detail how the problem-specific GA operators we designed for the GASP strategy work. After the individuals of the initial population are determined and their fitness is evaluated, the crossover, mutation, and selection operators are applied to drive the search in the right direction.

Crossover. The first step for the crossover operation is to select which individuals are chosen as mating partners. Although there are many known methods with respective benefits, we used the tournament and random variants. The tournament method selects two individuals randomly from the current population. Then it compares their fitness values, i.e., in our case their maximum worst-case delays, and chooses the one with the better value. The same procedure is repeated for the other mating partner. The random method is even simpler: two mating partners are selected randomly from the current population. Note that an individual might be selected more than once for a crossover operation. We experimented with both methods and found the random method to be more effective in our problem setting, probably because otherwise the explorative character of the search is lost too quickly. Once two individuals are selected the actual crossover operation can take place in order to produce new individuals by recombination of the parents. The crossover operation is at the very heart of a GA-based search and distinguishes it from other search techniques. It is very important for the crossover operation when recombining the chromosomes of the

parent individuals that building blocks are not destroyed with high probability. Under these general guidelines we designed the crossover operation as follows: As mentioned above, each individual is represented by an ordered set of indices, one index value for each sink. Now we choose a random start position in that list and choose a random number of positions such that from the starting point the chosen number of indices are exchanged between the individuals (see Figure 3 for an illustration). In doing so we make sure that this operation does not wrap around. This procedure is similar to the standard 2-point crossover with binary encoded chromosomes, but makes sure that indices are not cut through at intermediate positions during crossover. Furthermore, and more importantly, in most cases geographically close sinks remain together because they tend to be near each other on the chosen chromosome encoding. This interplay between chromosome encoding and crossover seemed to be inevitable to satisfy the building block hypothesis, because when we experimented with other crossover operators and chromosome encodings we achieved significantly less satisfying results

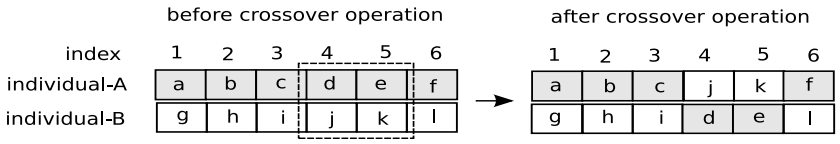

Fig. 3. Illustration of the crossover operation.

Mutation. Although the crossover operation is generally considered the more important operator of a GA, the mutation also plays an important role in the search process, since it does not only recombine existing solutions but tries to find new interesting areas of the search space. Especially with the above crossover operator mutation is indispensable since otherwise sink locations different from the one in the initial population would never be tried. The mutation operator we implemented works as follows: For each individual and each sink location (represented on the chromosome by the respective index) Algorithm 2 is applied. First it is checked whether a mutation should be applied or not, this is controlled by the mutation probability. This probability should not be too high because otherwise good partial solutions are destroyed too often by the mutation. However, in our case we use a relatively high mutation probability because the crossover operation is a pure recombination as already mentioned. Again by experimentation, we found a value of 0.4 for the mutation probability to deliver the best results in our scenarios. Now, the actual mutation operation consists of moving the index and thus the sink location up or down (with equal probability) in index space. This movement is restricted to a relatively small, compared to the index space, random number (less than 1% of the index space). This, in fact, weakens the disruptive nature of the high mutation probability we apply. Again, we experimented with different versions of mutation operators and only present the one that we found to deliver good results.

Algorithm 2 Mutation algorithm.

1. for a given index/sink location: choose random number r from [0,1]
2. if $(r > 0.8)$ then sink is moved up a small random number
   else if$(0.6 < r < 0.8)$ then sink is moved up/down a small random number
   else sink remains at its position

3 Performance Evaluation of GASP

This section describes the performance evaluation of the GASP strategy. For this purpose, we introduce two competitors for the sink placement problem under known sensor node locations

1. an exhaustive search of the space of possible combinations of candidate locations for the sinks to be placed, which we call Optimum Sink Placement (OSP) from now on, as it ensures to deliver the sink placement with minimal maximum worst-case delay for a given WSN;
2. a Monte Carlo-based search, from now on just called Monte Carlo Placement (MCP), where a certain number of purely random sink placements are generated and evaluated with respect to their maximum worst-case delay.

On the one hand, the OSP strategy can be considered as an upper bound on the performance that can be achieved by the GASP and will be used in order to demonstrate that at least for small-scale problems the GASP achieves a very good approximation of the optimal sink placement. The MCP, on the other hand, can be considered as a lower bound on what can be expected from the GASP. In comparison with the GASP, we always make sure that the MCP evaluates the same number of candidate sink placements. So, the comparison between the two is fair and as the computational effort is absolutely dominated by the sensor network calculus computations for the maximum worst-case delays of a given sink placement, the GASP and MCP have to invest nearly the same computational effort.

There are a number of factors for all the experiments conducted below: node distribution, field shape, routing strategy, duty cycle, and the number of nodes and sinks. Of course, we use the same node distribution for all strategies, a uniform random distribution over the sensor field. For the large scale WSNs, GASP is tested in 10 different node distributions and the average worst-case delays are compared with MCP. We assume a circular field shape with a size of $10000m^2$. The routing algorithm we use is based on Dijkstra's shortest path algorithm: nodes send their data to the sink with the shortest distance in hops (with ties being broken arbitrarily). The network size is varied from 30 to 500 nodes with a maximum of 7 sinks. Moreover, we use a transmission range of 16m for all sensor nodes. For the sensor network calculus computations, the popular token-bucket arrival curves and rate-latency service curves are considered for homogeneous nodes (see [10] for details). In particular, for the service curves we used rate-

latency functions which correspond to duty cylces of 1% and 5.61%[2] depending on the network size, since for larger networks a duty cycle of 1% resulted in infinite delay bounds. With respect to the GASP, we consider population sizes of 40 and 100 and the number of generations varies from 25 to 400, both of these parameters depending again on the network size. For the MCP, we generate an according number of random candidate sink placements. For each candidate sink placement a total flow analysis [11] for each node's traffic is performed in order to compute the maximum worst-case delay for the whole sensor network.

In the next two subsections, we first compare in small-scale networks how far the GASP deviates from the OSP and then look at its performance in large-scale networks with MCP as a lower bound benchmark.

3.1 Small-Scale WSNs: Comparison Between OSP and GASP

In this set of experiments, we analyze the worst-case delay for GASP and OSP for different, but relatively small network sizes of 30, 50, and 100 nodes. The number of candidate locations for the sinks were about 450, 950, and 2200 for the 30-, 50- and 100-node network, respectively. The number of sinks we had to place was restricted to 2 sinks, since for the 100-node network this already meant that the OSP strategy had to evaluate $\binom{2200}{2} = 2418900$ different combinations of sink placements where each evaluation consists of 100 total flow analyses which are not simple operations, resulting in a run-time of several days on a typical PC. For the GASP, on the other hand we chose a population size of 40 individuals and the number of generations was set to 200, resulting in only 8000 different sink placements that where evaluated in a few minutes. The worst-case delay results for the best sink placements found by GASP and OSP are shown in Figure 4.

As can be observed, the GASP performs very well in comparison to OSP: for the 30- and 50-node network it actually finds the global optimum, only for the 100 node case the best sink placement found by the GASP lies slightly above the one found by OSP (8.02s vs. 7.70s). That should be considered a success of the GASP since with a computational effort that is several orders of magnitude lower than for OSP, it achieves almost the same quality of solutions. Whether this holds true for larger-scale scenarios is difficult to assess as the OSP is prohibitively computationally expensive. Therefore, in the next subsection, we compare the GASP against the MCP in larger-scale networks.

3.2 Large-Scale WSNs: Comparison Between MCP and GASP

The purpose of this set of experiments is to assess the performance of the GASP strategy in larger WSN scenarios. The question we address is whether the GASP constitutes a more intelligent search strategy than a pure random search like MCP. In fact, there would even be the possibility that MCP could outperform

[2] The values are based on a realistic node model of a Mica2 mote running the TinyOS system (see CC1000 Radio Stack Manual) [1].

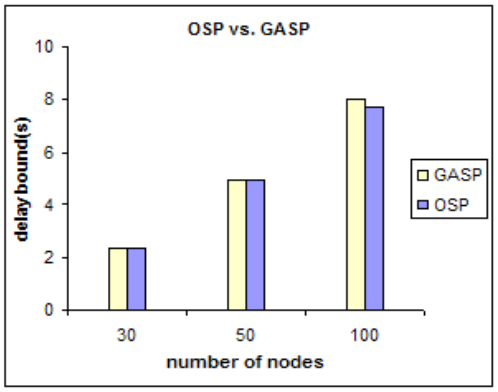

Fig. 4. The worst case delay comparison of OSP vs. GASP.

GASP. This would be the case if the GA operators were poorly designed and would mislead the GASP in areas of the search space that are fruitless. So, in a certain sense the following experiments also validate the design of the GASP operators.

We investigate a 500-node network with up to 7 sinks for 10 different scenarios, i.e., 10 different node distributions. For each of the scenarios, this resulted in approximately 13000 candidate sink locations, so that at maximum the search space becomes as big as $\binom{13000}{7} \approx 1.24 \times 10^{25}$. On the other hand for the GASP we used a population size of 100 with 100 generations until termination resulting in 10000 sink placements being evaluated for their worst-case delay. As mentioned above, we allow the same amount of evaluations to the MCP. In any case, it is clear that this amount of sink placement evaluations constitutes only a tiny fraction of the overall search space.

The results (averaged over the 10 scenarios) of these experiments are shown in Figure 5. This analysis shows that GASP performs better than MCP. For the GASP strategy, the worst-case delay improves from from 5.7 to 4.1 to 3.2 to 2.7 to 2.3 to 2.0 for the 2- to 7-sink scenarios, respectively. For the MCP, the worst-case delay improves from 6.6 to 5.0 to 4.2 to 3.3 to 3.2 to 2.5 for the 2- to 7-sink scenarios, respectively. Morevover, at a confidence level of 90%, the confidence intervals of both strategies do not overlap for the respective number of sinks, which shows the statististical significance of the results. The delay difference between the two strategies is roughly about 1s for most cases. One may think that this is not spectacular, but it should be kept in mind that we are dealing with bounds which usually cannot be improved by factors. Furthermore, we want to emphasize once more that the computational effort for GASP and MCP is approximately the same (with the GASP exhibiting a run-time penalty being in the fraction of a percent of the MCP's run-time). This shows that the GA operators do something sensible as the GA search improves on the pure random search of the MCP.

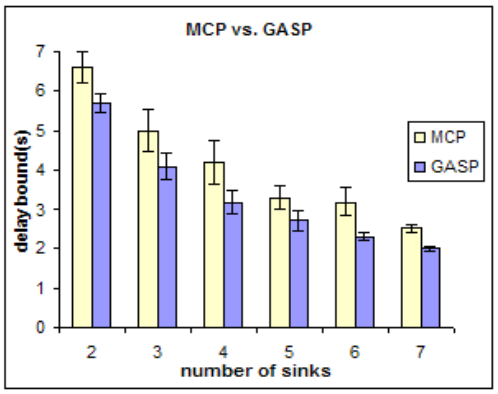

Fig. 5. The worst case delay comparison of MCP vs. GASP.

4 Related Work

Although being an obviously very practical issue for WSNs, the placement of multiple sinks has been addressed surprisingly seldom as a WSN design problem in the literature. In [8] and [6], sinks are placed using a cluster-based model. Both of them suggest the cluster-head as an interesting point to place sinks and thus the sink location should be chosen from the set of cluster heads. So, besides being constrained on cluster-based WSNs, they also focus on energy-efficiency as optimization criterion whereas we focus on the delay while achieving energy-efficiency goals by setting duty cycles accordingly. Another somewhat related work can be found in [13], where the problem of optimally steering mobile sinks is approached by using an electrostatic model for maximizing the lifetime of the WSN. In a similar vein, [2] discusses how mobile sinks can be effectively controlled. Our work focuses on stationary sink placement, although we could envision for future work to make our GASP strategy fast enough to work well also for the on-line problem of steering mobile sinks. From the perspective of sensor network calculus, [10] and [11] are basic foundations for the worst-case analysis on which this study is built. In [12] it is demonstrated how these foundations can be extended to the multiple sink case, yet no attempt is made to optimize the sink placement.

5 Conclusion

In this paper, we addressed the issue of placing multiple sinks in a time-sensitive WSN such that the worst-case message transfer delay is minimized. First, we provided a method to discretize the search space for this optimization problem without losing any information and thus without harming the optimality of solutions. This discretization allows to apply classical search techniques. In principle,

an exhaustive search can be performed to find the globally optimal sink placement. Yet, as was demonstrated, this is computationally infeasible and, therefore, we developed a GA-based search technique called GASP. The performance evaluation of GASP showed that for small-scale networks, the GASP strategy is very close to the performance of an exhaustive search. For large-scale networks, it was shown that the performance of GASP is better than a pure random search, thus validating the design of the problem-specific GA operations. In conclusion, the GASP strategy seems to constitute a good heuristic for a near-optimal sink placement in time-sensitive WSN applications.

References

1. http://www.tinyos.net/tinyos-1.x/doc/mica2radio/cc1000.html.
2. C. Chen, J. Ma, and K. Yu. Designing Energy-Efficient Wireless Sensor Networks with Mobile Sinks. In *Proceeding of the 4th ACM Conference on Embedded Networked Sensor Systems (SenSys 2006)*, Boulder, Colorado, USA., October 2006.
3. David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Kluwer Academic Publishers, Boston, MA, 1989.
4. John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.
5. H. Karl and A. Wittig. *Protocols and Architectures for Wireless Sensor Networks.* Wiley, 2005.
6. H. Kim, Y. Seok, N. Choi, Y. Choi, and T. Kwon. Optimal Multi-sink Positioning and Energy-efficient Routing in Wireless Sensor Networks. In *Lecture Notes in Computer Science(LNCS)*, volume 3391, pages 264–274, 2005.
7. J.-Y. Le Boudec and P. Thiran. Network Calculus - A Theory of Deterministic Queuing Systems for the Internet. In *Lecture Notes in Computer Science*. Springer, 2001.
8. E. I. Oyman and C. Ersoy. Multiple sink network design problem in large scale wireless networks. In *IEEE International Conference on Communications (ICC)*, volume 6, pages 3663–3667, June 2004.
9. Jens B. Schmitt and Wint Yi Poe. Minimizing the Maximum Delay in Wireless Sensor Networks by Intelligent Sink Placements. Technical Report 362/07, University of Kaiserslautern, Germany, July 2007.
10. Jens B. Schmitt and Utz Roedig. Sensor Network Calculus - A Framework for Worst Case Analysis. In *Proceedings of the International Conference on Distributed Computing in Sensor System (DCOSS05)*, June 2005.
11. Jens B. Schmitt and Frank A. Zdarsky. The DISCO Network Calculator - A Toolbox for Worst Case Analysis. In *Proceeding of the First International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS'06)*. ACM, November 2006.
12. Jens B. Schmitt, Frank A. Zdarsky, and Utz Roedig. Sensor Network Calculus with Multiple Sinks. In *Proceedings of the Performance Control in Wireless Sensor Networks Workshop at the 2006 IFIP Networking Conference*, pages 6–13, May 2006.
13. Zoltan Vincze, Kristof Fodor, Rolland Vida, and Attila Vidacs. Electrostatic Modelling of Multiple Mobile Sinks in Wireless Sensor Networks. In *Proceedings of the Performance Control in Wireless Sensor Networks Workshop at the 2006 IFIP Networking Conference*, May 2006.