Particle Swarm Optimization Based Approach to Solve the Multiple Sink Placement Problem in WSNs

Haidar Safa, Wassim El-Hajj, Hanan Zoubian
Computer Science Department
American University of Beirut
Beirut, Lebanon
{haidar.safa, wassim.el-hajj, hgz01 }@aub.edu.lb

Abstract—**A wireless sensor network (WSN) is a collection of tiny and limited-capability sensor nodes that report their sensed data to a data collector, referred to as a sink node. WSNs are used in many applications, but are challenged by memory and energy constraints. To address these issues, solutions have been proposed on different levels including the topological level where multiple sinks can be used in the network to reduce the number of hops between a sensor and its sink node. Topological level solutions are very crucial in time-sensitive applications where the maximum worst case delay incurred by a message to get from a sensor to the corresponding sink should be minimal or at least less than a certain value. In turn, the maximum worst case delay can be minimized by choosing near optimal locations of the sinks. Consequently the network lifetime will be extended since the energy consumed by the sensor nodes will be reduced. In this paper, we propose an efficient and robust approach based on Particle Swarm Optimization (PSO) heuristic to solve the multiple sink placement problem; more specifically we use Discrete PSO (DPSO) with local search (LS). We start by formulating the problem then discretizing it and finally applying PSO while introducing local search to the inner workings of the algorithm. When compared to Genetic Algorithm-based Sink Placement (GASP), which is considered the state-of-the-art in solving the multiple sink placement problem, our approach improved the results in most scenarios while requiring less runtime.**

Keywords: Wireless Sensor Networks, Sink Placement Problem, Particle Swarm Optimization, Genetic Algorithms

I. Introduction

Wireless Sensor Networks (WSNs) are currently widely used in military applications, civil applications, habitat monitoring, and environmental observations. WSNs are composed of low-cost, low-power, multifunctional, and tiny sensor nodes that collect physical information, and forward them to sink nodes which act as data collectors. Sensors can be deployed densely and randomly in a certain area without predetermining their positions. The fact that requires them to exhibit self-configuring and self-organizing capabilities.

The biggest challenge in WSNs remains to be the limited energy of the sensor nodes because sensor nodes act as data sources and routers at the same time. Moreover, the traffic load on the various sensors is not balanced since nodes near the sink get more depleted than others. To improve the use of available resources and increase the lifetime of WSNs, most researcher adapted approaches that aim to reduce the number of transmitted messages in the network [3-5, 20]. Such approaches include routing [6, 8-10], data reduction [7], event filtering [11, 12], load balancing, energy efficient circuitry, and topological control & planning [5]. In this paper, we suggest a technique that falls under the topological planning area where a certain number of sinks is added to the network for the purpose of making the network more manageable and prolonging its lifetime by reducing the energy dissipated at each sensor node. This problem is referred to as: multiple sink placement problem. Unfortunately, finding the appropriate number of sinks and their best locations is an NP-Hard problem, so no exact solution can be found for large WSNs. A good approximation algorithm can still produce good results leading to (1) better load balancing, (2) fewer number of intermediate hops needed to reach the closest sink, (3) less delay incurred by a message from a sensor to the corresponding sink, and (4) reduced amount of utilized resources. All these advantages translate to increased network lifetime.

In this paper, we use an enhanced version of the Particle Swarm Optimization (PSO) to solve the sink placement problem in WSN. This enhanced version, Discrete PSO (DPSO) with local search (LS), is composed of (1) discretizing the sink placement problem, (2) using local search, and (3) applying PSO to find a solution. Our aim is to find the best combination of sink locations that minimizes the fitness function defined as "the maximum worst case delay in the network". As shown later in the section V, our proposed approach outperforms a state of the art algorithm called Genetic Algorithm-based Sink Placement (GASP) [1, 2], which is usually used to solve the sink placement problem.

PSO is a heuristic based on artificial life and evolutionary computing. Its behavior is different from pure evolutionary computing methods such as genetic algorithms (GA) and Evolutionary Simulated Annealing (ESA) and according to [27], it even performs better. In swarm intelligence (SI) [26, 28], the strategy changed from being based on competition as in GA and the focus became on a social model of optimization. If a particle's (individual's) neighbor has a better performance for a problem [25], the particle tries to imitate its neighbor. Basically, the individuals create a working social network where they collaborate together to get to a better result.

The rest of the paper is structured as follows. In section II, we discuss the approaches suggested in literature to solve the sink placement problem. In section III, we present the analytical details needed to calculate the fitness function which is used as our evaluation metric. In section IV, we detail our approach which solves the Sink Placement Problem using DPSO with LS. In section V, we present the experimental results and show how our approach outperforms GASP. In section VI, we conclude the paper and summarize our contribution.

II. Related Work

Various solutions were proposed in the literature to solve the multiple sink placement problem. In [23], the paper assumes that the WSN can be deployed in an agricultural field where the nodes are distributed in a nearly uniformed way where the sensors can be dropped from a plane or placed manually. To ensure scalability, the WSN is divided into clusters where each cluster has its own sink to which all the nodes in that cluster report to. Three sink placement schemes were proposed each of which addresses the problem from a different angle: "Best Sink Location (BSL)", "Minimizing the number of Sinks for a Predefined minimum Operation Period (MSPOP)", and "Finding the Minimum number of Sinks while Maximizing the Network Lifetime (MSMNL)". In BSL, k-means clustering is used to find the sink locations. In MSPOP, a brute force technique is used starting with 1 sink and incrementing the number of sinks, while evaluating the network lifetime. The solution is obtained when the given constraint is reached. In MSMNL, the problem is formulated such that the network lifetime is maximized subject to the most economical investment, but no simulations were conducted.

In [24], the aim was to extend the network lifetime by distributing the load among the deployed sensors. The problem was formulated to improve the lifetime and fairness of the WSN with multiple sinks. A clustered WSN architecture is assumed where the cluster head has forwarding functionality. Two problems were solved: the location of sink nodes and the traffic flow rates of the routing paths. Both problems were formulated differently using Linear Programming (LP) and solved using CPLEX (an Integer LP solver). The main problem with this method is its failure to run for large WSNs.

In [1,2], several sink placement strategies were presented, discussed and evaluated to achieve the goal of minimizing the maximum worst-case delay and maximizing the lifetime of a WSN. These sink placement strategies are: random sink placement (RSP), geographic sink placement (GSP), intelligent sink placement (ISP) and the genetic algorithm-based sink placement (GASP). GASP was their proposed algorithm that performed the best under realistic scenarios. For that reason, we chose GASP to compare our method to. As will be shown later in section V, our proposed method performs better than GASP in the sense that it picks better locations for the sinks and moreover it requires less time to execute.

III. Fitness Function Calculation

As mentioned earlier, our aim in this paper is to find the best combination of sink locations that minimizes the fitness function defined as "the maximum worst case delay in the network". Given a WSN, we find the fitness function using Wireless Sensor Networks Calculus and one of its main supporting tools known as the DISCO network calculator [15]. In what follows, we overview the WSN concepts needed to formulate the problem and calculate the fitness function. We also introduce WSN calculus and the inner workings of DISCO network calculator.

WSN nodes are characterized by their sensing rate which is defined by the application. For instance, if a sensor sends periodically 1 message every 30 seconds to the nearest sink and every packet message is 288 bits, then the sensing rate of the sensor is 9 bits/second. In addition, the transceiver in the node alternates between two states: active (transmit, idle, receive) and sleep. The *duty cycle* represents the fraction of time where the sensor node is in an active state (a.k.a. active time/total time). It is defined in the application of the nodes and corresponds to a certain *forwarding rate* and *forwarding latency* (due to the sleep and idle time) in the network. A smaller duty cycle means that the node's active time is decreased, and the sleep time is increased. This has the advantage of saving energy in the node, since a node consumes much more energy in the active state. In the MICA2-CC1000 [21] sensor node that we use in this paper, the energy consumed by the node's transceiver operating at 868/916 MHz is less than 1 μA in the sleep state, in comparison with 10 mA in the receive and 27 mA in the transmit with maximum power.

Sensor network calculus is an analytical framework for worst-case analysis in WSNs [13, 14]. The sensor network calculus is used to get maximum message transfer delays, maximum buffer requirements and lower bounds on duty cycles. The DISCO network calculator was proposed as an implementation of the sensor network calculus [15, 18]. It calculates the network delay and throughput, taking into account a worst-case analysis instead of the average one. It uses curves to describe the network traffic and services by using min-plus (R, min, +) algebra. These curves are increasing functions that represent accumulated data. In addition, compression, aggregation and other processing techniques can be used to cause data scaling (i.e. data is not just forwarded but also mutated along the path). DISCO network calculator was used by many researches [1-2, 19-21] and was proven to have very close results to those obtained by actual simulations [19].

To use DISCO network calculator [18], we used a topology generator program called BRITE [17] to generate different topologies of sensor nodes with different sizes. Once the network topology is modeled, the basic input and output of each node is described by an *arrival curve* and a *service curve* as explained later. Then, a flow is created between every source node and its corresponding sink.

To analyze the flows to the sink, several methods can be used such as fair queuing analysis (FQA), total flow analysis (TFA), separated flow analysis, and PMOO analysis [22]. In this paper, TFA is used to get the worst case delay bound for every flow of interest in the network. It is to be noted that we formulated an algorithm that enabled us to use TFA for multiple sink scenarios, but the details of this algorithm are very mathematical and out of the paper scope. We define next the arrival and service curves as they are needed later.

A. Arrival Curve

The sensing arrival curve represents the basic input of each sensor node in WSN. Each sensor node *i* has its own arrival curve as shown in equation (1). The arrival curve $\bar{\alpha}_i$ at node *i* is basically the sum of the sensed input traffic (α_i) and all received sensed data $\alpha^*_{\text{child }(i,j)}$ of its child nodes j=1 to n_i (with n_i being the number of node's *i* children, if *i* is not a leaf node). Sensor node *i* processes its input and then sends it to its parent node. The arrival curve for the input at node *i* is given as:

$$\bar{\alpha}_i = \alpha_i + \sum_{j=1}^{n_i} \alpha^*_{\text{child }(i,j)} \tag{1}$$

Where $\bar{\alpha}_i$ is the arrival curve at node *i*, α_i is the sensed input traffic at node *i*, and α^*_i represents the forwarded data from node's *i* children.

B. Service Curve

The service curve depends on the way packets are scheduled in a sensor node which depends on the transceiver speed, the chosen link layer protocol and the duty cycle. It mainly depends on how the energy-efficiency goals are set [13] and whether aggregation is used in the network. The service curve equation is given as follows:

$$\beta_i(t) = \beta_{f_i, l_i}(t) = f_i\,[t - l_i]^+ \tag{2}$$

Where β_i is the service curve, f_i is the forwarding rate, and l_i is the forwarding latency for sensor node *i*.

Given a certain duty cycle % assigned to the nodes, the forwarding rate and latency can be calculated. The maximum forwarding latency l_i is basically the sleep time plus the idle time. It is the maximum time a message has to wait until the forwarding capacity is available again.

IV. Sink Placement Problem Solution using DPSO with LS

We propose to use DPSO with local search to solve the sink placement problem. The objective function in the sink placement problem is to minimize the maximum worst case delay. This problem can be formulated as follows:

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

where

$$d_i = f(\mathcal{T} \mid \alpha, \beta)$$
$$\mathcal{T} = g(s \mid p, \mathcal{R})$$
$$p = \left(p_i^{(x)}, p_i^{(y)}\right)_{i=1,\ldots,n}$$
$$s = \left(s_i^{(x)}, s_i^{(y)}\right)_{j=1,\ldots,k}$$
$$p_i, s_i \in \mathcal{F}$$

d_i is the worst case delay for sensor node i, $\mathcal{T}$ is the topology of the sensor network, α is the arrival curve, β is the service curve, s is the vector containing the actual decision variables (i.e. the locations chosen from the candidate locations of the k sinks), p is the vector containing the locations of the n sensor nodes (given by the problem), $\mathcal{R}$ is the routing algorithm (given by the problem), and $\mathcal{F}$ is the sensor field (a square field).

The worst case delay d_i for sensor node i is a function of the topology of the sensor network field $\mathcal{F}$, the arrival curve α and the service curve β of sensor node i. The topology is a function of the location of the sensors, location of the sinks, and the routing algorithm $\mathcal{R}$ given by the problem. Therefore, the location of the sinks affects the worst case delay d_i. In this problem, it is assumed that the locations of the n sensors and the used routing algorithm are given. The only decision variables are the locations of the k sinks. Since this problem is a continuous optimization problem, where d_i function is highly non-linear, we use DPSO with local search to find a feasible solution. Algorithm 1 outlines the pseudocode of our approach.

```
Algorithm 1: Multiple Sink Placement Alg. using DPSO with LS
Begin
  Initialize sinks locations vector from the list of candidate locations
  Calculate fitness value of vector $Y_i$ using the DISCO calculator
  Do{
        Find personal best ($P_i^t$)
        Find global best ($G^t$)
        For each particle
              - Apply velocity component (equation 4)
              - Apply cognition component (equation 5)
              - Apply social component (equation 6)
              - Calculate fitness value of vector $Y_i^t$ using the
                DISCO calculator
        End
        Apply local search (for $DPSO_{LS}$) to global best
  } while (maximum iteration is not reached)
End
```

where

$$Y_i^t = c_2 \oplus F_3\left(c_1 \oplus F_2\left(w \oplus F_1(Y_i^{t-1}), P_i^{t-1}\right), G^{t-1}\right) \tag{3}$$
$$\lambda_i^t = w \oplus F_1(Y_i^{t-1}) \tag{4}$$
$$\delta_i^t = c_1 \oplus F_2(\lambda_i^t, P_i^{t-1}) \tag{5}$$
$$X_i^t = c_2 \oplus F_3(\delta_i^t, G^{t-1}) \tag{6}$$

After initializing a population of particles, at every generation, an exchange operator is applied where two distinct locations are picked randomly and swapped with a probability *w*, as shown in equation (4). Then, in equation (5), one-cut crossover with a probability of c_1 is applied to all particles *i*, with λ_i^t as the first parent and P_i^{t-1}(the personal best of particle i from the previous generation) as the second parent, and one of the two children is selected randomly. This equation represents the cognition part δ_i^t. Next, in equation (6), two-cut crossover with a probability of c_2 is applied, with δ_i^t as the first parent and G^{t-1} (the global best of all particles from the previous generation) as the second parent, and one of the two children is selected randomly. This equation represents the social part X_i^t. In summary, at every generation, new possible solutions are explored by each particle by changing its values, and trying to move closer to the personal best and the global best. The local search is done by changing the value of one of the sinks locations in the global best i.e. picking a new location from the list of candidate locations. The methodology used to find the pool of candidate locations is discussed next.

A "region of indifference" is defined to be a region where a sink can be placed anywhere inside it without a change in the routing topology and thus without a change in the maximum worst case delay. Each region can have one candidate location representing it. Given the location of sensor nodes and their transmission ranges, the total number of regions of indifference (i.e. candidate locations) is calculated using this equation:

$$N_{(n)} = \left(\sum_{i=1}^{n-1} 2 * nb_i\right) + 1 \tag{7}$$

where n is the number of sensor nodes and nb_i is the number of neighboring nodes after the elimination of nodes $1,\ldots,i-1$. However, the exact number of regions of indifference can be different than the number calculated by equation 7 because some areas are eliminated due to some cyclic dependencies, scattered indifference regions, or common intersection points.

After getting the number $N_{(n)}$ which represents an upper bound on the number of candidate locations, getting the actual candidate locations is done by discretizing the search space [1-2]. The discretization proceeds as follows. After laying a grid over the sensor field, we check for each grid point whether the distance between the candidate position on the grid and the node position is strictly less than the transmission range of the node (i.e. determining the neighborhood of the grid point in terms of the transmission areas in which it is contained) and the distance of the candidate position to the origin is less than the radius. If yes, we add this candidate position to the candidate locations as shown in algorithm 2.

```
Algorithm 2: Candidate Locations
1. read sensor nodes positions
2. calculate total number of regions ($N_{(n)}$) from equation 7
3. adjust sampling granularity and map a candidate location for each
   region:
   forall (n)
       if(candidatePosToNodePos < txRange &&
                   originToCandidatePos < radius)
           candidate locations ++
```

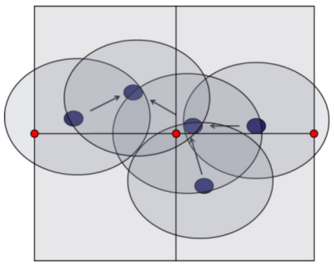
Figure 1. Laying a grid with certain granularity

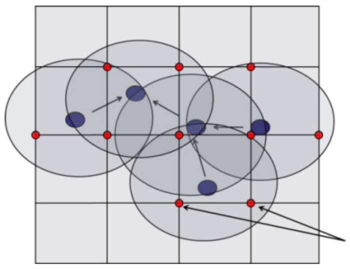
Figure 2. Laying a grid with increased granularity

Then, we merge the grid points which have identical neighborhoods by choosing only one of them as a representative node for that particular region of indifference in which they belong to. The number of acceptable grid points is then compared with the number $N_{(n)}$ previously calculated. If that number is still significantly less than $N_{(n)}$, then the granularity is increased as shown in Figs. 1 and 2. By decreasing the cell size in the grid, we get more grid points to check. This process gets repeated until the sampling accuracy is high enough i.e. all or most of the regions of indifference have been identified and each region is represented by a grid point (i.e. sink candidate location).

V. EXPERIMENTAL RESULTS

In this section, we evaluate the performance of our approach (DPSO) and GASP algorithms when used to solve the sink placement problem. We describe the performance evaluation environment model and parameters used. Then we present the experimental results and analyze them while varying the number of sinks, duty cycle, and other parameters.

The simulation environment is a topology of N sensor nodes that communicate in an ad hoc fashion to send messages to the nearest sink. The application used is time-critical, meaning that messages should get to the nearest sink with a delay strictly less than a certain value. Our objective therefore becomes to calculate and minimize the maximum worst case delay incurred by any message to get to the nearest sink.

In order to minimize the maximum worst case delay in the WSN, GASP and DPSO algorithms are used to find near optimal solutions for sink placement. Given the topology, the number of sensors and their positions, along with the corresponding candidate list and candidate list neighbors (calculated using the method discussed in section IV), the algorithms initialize the first population of solutions. The fitness is then calculated by DISCO network calculator and returned to GASP or DPSO, and decisions are made about what combinations of sink locations will be used in the next generation of solutions.

In all the simulations, sensor nodes are placed randomly or in a heavy-tailed distribution in an area size of 10000 m^2. The reported results are the average of 5 runs. The sensor nodes are MICA2 sensors [21] with a data rate of 19.2 kbps and a transmission range of 16 meters. Sensors and sinks are assumed to be stationary (not mobile), and the routing protocol used is a shortest path based protocol, where each sensor reports its data to the nearest sink. Sinks are ultimately placed according to GASP or DPSO. Table 1 presents a summary of the parameters used in the simulations along with their values. Tables 2 shows the effect of the assigned duty cycle on the effective throughput f and the maximum forwarding latency l, which are needed as parameters for the service curve.

TABLE 1. SIMULATION PARAMETERS AND THEIR VALUES

Parameter	Value
Number of sensor nodes, n	100, 200, 500 nodes (homogeneous)
Number of sink nodes, s	2, 3, 4, 5, 6, 7, 8 sink nodes
Topography	100 x 100 meters
Distribution	Random, heavy-tailed
Routing protocol	Shortest Path based protocol
Transmission range	16 meters
Packet length	36 bytes (MICA 2)
Data rate	19.2 kbps (MICA 2)
Arrival curve	Token bucket
Sensing Rate	36 byte TinyOS packet every 30 seconds
Service curve	Rate latency
Duty cycle	1 %, 2.22 %
Simulation Time	30 min, 1 hr. 30 min, 5 hrs, 12 hrs, 90 hrs.

TABLE 2. THE EFFECT OF THE DUTY CYCLE ON THE MICA2 PROPERTIES

Duty cycle (%)	**Max packets (packets/s)**	**Effective throughput (kbps)**	**Preamble length (bytes)**	**Sleep Time T2 (ms)**	**Max forwarding latency (l) (ms)**
100	42.93	12.364	28	0	11
35.5	19.69	5.671	94	20	31
11.5	8.64	2.488	250	85	96
7.53	6.03	1.737	371	135	146
5.61	4.64	1.336	490	185	196
2.22	1.94	0.559	1212	485	496
1.00	0.89	0.258	2654	1085	1096

As shown in table 2, for the duty cycle $\delta = 1\%$, the effective throughput f is equal to 0.258 kbps (a.k.a. 258 bits per second) and the latency l is equal to 1.096 s. Using equation 2, the rate-latency service curve used in DISCO network calculator can be derived as follows: $\beta_i(t) = \beta_{f_i,l_i}(t) = f_i\,[t - l_i]^+ = 258\,(t - 1.096)+$ [bit]. Whereas, for the duty cycle $\delta = 2.22\%$, the following rate-latency service curve is used: $\beta_i(t) = \beta_{f_i,l_i}(t) = f_i\,[t - l_i]^+ = 559\,(t - 0.496) +$ [bit]. Tables 3 and 4 present the parameters used for GASP and DPSO.

TABLE 3. GASP PARAMETERS

Parameter	Value
Population size	14, 40, 100 chromosomes
Max. number of generations	25, 40 generations
Parent Selection	Random method
Mutation probability r	0.4

TABLE 4. DPSO PARAMETERS

Parameter	Value
Swarm size	40, 100 individuals
Max. number of generations	25, 40 generations
Cognitive probability c1	0.5
Social probability c2	0.5
Inertia Weight w	0.9

Two metrics are calculated to evaluate the performance:

1. Maximum worst case delay which is defined as the maximum worst case delay incurred by the whole network.
2. Execution time that is incurred by the two algorithms while increasing the number of nodes and the number of sinks.

We ran the algorithms on an Intel processor Core2 Quad CPU, 2.33 GHz and 3.21 GB of RAM. Before applying GASP or DPSO, we find the initial pool of candidate locations. For 100, 200, and 500 randomly generated nodes, the average num of candidate locations was 1490, 4074, and 8075 respectively. When heavy tailed distribution is used, the number of candidate locations increased by almost 15%.

In the first set of experiments, we compared the execution time of GASP and DPSO on all random topologies (100, 200 and 500 nodes), while varying the population size and the number of generations. The results show a significant advantage of DPSO over GASP as the execution time of DPSO was much less than GASP while obtaining similar or even better sinks' placement. For instance, for 100 nodes topology, the execution time of GASP having 100 chromosomes and 40 generations was almost 4 hours, while the execution time of DPSO having 100 particles and 40 generations was almost $1\frac{1}{2}$ hours. Similar results were obtained for larger topologies. This is due to the fact that when using GASP, in every generation, the initial population of *P* combinations does crossover and mutation producing *3P* new combinations, each of which will be evaluated for their fitness. Whereas, in DPSO in every new generation, the initial population of *P* combinations just change its values without producing any new populations and thus only P fitness evaluations will be performed.

In the second set of experiments, we compared the maximum worst case delay (fitness function) when GASP and DPSO are used. The delay is calculated using the sensor network calculus with the Total Flow Analysis (TFA) method. Under the assumptions of homogeneous sensor nodes sending a packet every 30 seconds to the nearest sink, the following values are used in the network calculus to calculate the worst case delay: (1) a token bucket arrival curve with sensing rate of 9 bits/second and delay of 0 second for all scenarios. (2) For a duty cycle of 1%, a forwarding rate of 258 bits/s and a forwarding latency of 1.096 seconds are incurred. Whereas a duty cycle of 2.22% has a forwarding rate of 559 bits/s and a forwarding latency of 0.496 second. (3) Routing from the node to the nearest sink is done using shortest path algorithm.

In the captions of the figures below, "GASP" is followed by the number of chromosomes and the number of generations. Similarly "DPSO" is followed by the number of individuals and the number of generations. Figs. 5, 6, and 7 show the maximum worst case delay obtained by GASP and DPSO when 100 nodes are deployed with 1% duty cycle while varying the method's parameters. DPSO produced results that are better than GASP. In few scenarios, GASP's performance was similar to DPSO or slightly better. Yet DPSO has another advantage of taking much less time than GASP since the number of evaluated solutions in DPSO is 1/3 that of GASP.

In Figs 8, 9 and 10 we vary the duty cycle and repeat the experiments. We set the duty cycle to be 2.22% instead of 1%, which translated to more active time, more processing, and more energy consumption, but less time to finish a job. DPSO continues to outperform GASP except in the case of 5 and 6 sinks in Fig. 10. The improvement gap varies between 0.17 and 0.86 second, which is a significant improvement given that 2.9 and 0.5 seconds are the maximum and minimum delays achieved, respectively. The savings are sometimes one third.

We did similar experiments on larger topologies composed of 200 and 500 sensor nodes. Figs. 11 and 12 show the results when 200 and 500 nodes are used, both with 1% duty cycle. In both scenarios, it is clear that DPSO outperformed GASP or at least achieved the same results. However, DPSO execution time was always less than GASP. Experiments on heavy tailed distributions also showed the power of DPSO over GASP.

VI. CONCLUSION

In this paper, we addressed the issue of energy limitations in WSNs by suggesting an efficient heuristic to solve the sink placement problem. Our heuristic was based on Discrete Particle Swarm Optimization (DPSO) with local search. First we formulated the problem and transformed it to a discrete problem by discretizing the search space and getting all possible candidate locations for sinks. Then we implemented and ran experiments using both DPSO and GASP algorithms on random and heavy-tailed topologies with different duty cycles and different number of sinks. Each solution's fitness (a.k.a. maximum worst case delay) is evaluated using the Total Flow Analysis (TFA). Finally, results from both algorithms are compared. In almost all experiments, DPSO succeeded to get better results than GASP. Moreover, DPSO with local search does not converge fast to a local optimal solution and still finds better solutions at advanced generation numbers. Concerning the execution time, DPSO always produced the results much faster than GASP, since GASP produces at each generation triple the number of populations produced in DPSO.

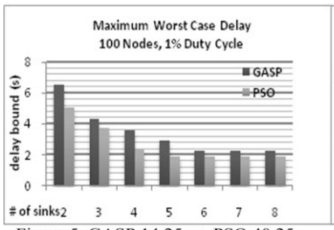

Figure 5. GASP 14 25 vs. PSO 40 25

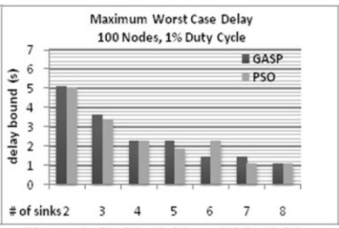

Figure 6. GASP 40 25 vs. PSO 40 25

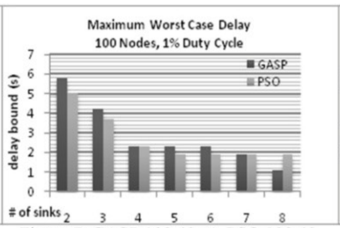

Figure 7. GASP 100 40 vs. PSO 100 40

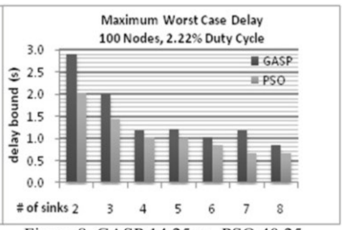

Figure 8. GASP 14 25 vs. PSO 40 25

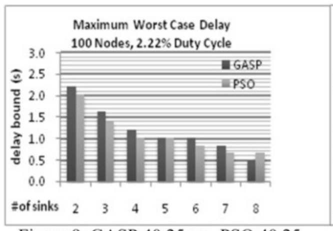

Figure 9. GASP 40 25 vs. PSO 40 25

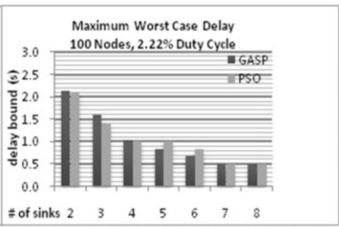

Figure 10. GASP 100 40 vs. PSO 100 40

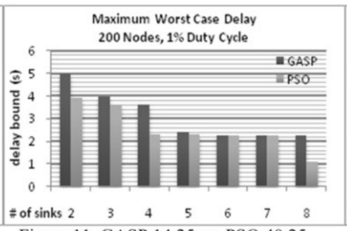

Figure 11. GASP 14 25 vs. PSO 40 25

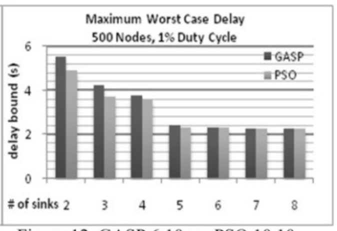

Figure 12. GASP 6 10 vs. PSO 10 10

REFERENCES

[1] Wint Yi Poe and Jens B. Schmitt, "Minimizing the Maximum Delay in Wireless Sensor Networks by Intelligent Sink Placement", Tech. Report No. 362/07, University of Kaiserslautern, Germany, July 2007, pp. 1-20

[2] Wint Yi Poe and Jens B. Schmitt, "Placing Multiple Sinks in Time-Sensitive Wireless Sensor Networks using a Genetic Algorithm", In Proceedings of the 14th GI/ITG Conference on Measurement, Modeling, and Evaluation of Computer and Communication Systems (MMB 2008), Dortmund, Germany, March 2008, pp. 253-268

[3] W. Ye, J. Heidemann and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks", Proc. of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002), New York, June 2002, pp. 1567-1576

[4] Ya Xu, John Heidemann and Deborah Estrin, "Geography-informed energy conservation for ad hoc routing", In Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom 2001), Rome, Italy, July 2001, pp. 70-84

[5] Ya Xu, Solomon Bien, Yutaka Mori, John Heidemann and Deborah Estrin, "Topology control protocols to conserve energy in wireless ad hoc networks", Technical Report 6, University of California, Los Angeles, Center for Embedded Networked Computing, January 2003.

[6] Benjie Chen, Kyle Jamieson, Hari Balakrishnan and Robert Morris, "SPAN: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks", In Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM 2001), July 2001, pp. 85-96

[7] Yong Yao, J. E. Gehrke, "The Cougar Approach to In-Network Query Processing in Sensor Networks", ACM Sigmod Record, vol. 31, no. 3, September 2002, pp.9-18

[8] Chalermek Intanagonwiwat, Ramesh Govindan and Deborah Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks", In International Conference on Mobile Computing and Networking (MOBICOM 2000), August 2000, pp. 56-67

[9] D. Braginsky and Deborah Estrin, "Rumor Routing Algorithm for Sensor Networks", In WSNA, Georgia, USA, Sep. 2002, pp. 20-31

[10] Brad Karp and H.T. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks", In MOBICOM 2000, pp. 243-254

[11] Ramesh Govindan, Joseph Hellerstein, Wei Hong, Samuel Madden, Michael Franklin and Scott Shenker, "The sensor network as a Database", Number TR02-02-771, September 2002

[12] S. Ratnasamy, D. Estrin, R. Govindan, B. Karp, S. Shenker, L. Yin and F. Yu, "Data-centric storage in sensornets", Proceedings of the First Workshop on Sensor Networks and Applications (WSNA), Atlanta, GA, Sept. 2002

[13] J. B. Schmitt and U. Roedig, "Sensor Network Calculus - A Framework for Worst Case Analysis", *Proceedings of IEEE/ACM International Conference on Distributed Computing in Sensor Systems (DCOSS'05)*, Marina del Rey, USA, June 2005, Springer, LNCS 3560, pp. 141-154.

[14] Jens B. Schmitt, Frank A. Zdarsky, and Utz Roedig, "Sensor Network Calculus with Multiple Sinks", *In the Proceedings of the Performance Control in Wireless Sensor Networks Workshop at the 2006 IFIP Networking Conference*, Coimbra, Portugal, May 2006, pp. 6-13.

[15] The DISCO Network Calculator, http://disco.informatik.unikl.de/content/Downloads, February 2009

[16] JUNG – the Java Universal Network/Graph Framework, http://jung.sourceforge.net/, January 2008

[17] Boston University Representative Internet Topology Generator (BRITE), http://www.cs.bu.edu/brite/, January 2008

[18] Nicos Gollan, Frank A. Zdarsky, Ivan Martinovic, and Jens B. Schmitt, "The DISCO Network Calculator", In Proceedings of the 14th GI/ITG Conference on Measurement, Modeling, and Evaluation of Computer and Communication Systems (MMB 2008), Dortmund, Germany, March 2008, pp. 291-294

[19] U. Roedig, N. Gollan and J. B. Schmitt, "Validating the Sensor Network Calculus by Simulations", *In Proceedings of the 2nd Performance Control in Wireless Sensor Networks Workshop at the 2007 WICON Conference*, Austin, Texas, USA, ACM Press, October 2007.

[20] N. Gollan and J. B. Schmitt, "Energy Efficient TDMA Design Under Real-Time Constraints in Wireless Sensor Networks", In Proceedings of the 15th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'07), IEEE, October 2007, pp. 80-87

[21] MICA2, http://www.cs.purdue.edu/homes/fahmy/software/iheed/tinyos-1.x/tos/platform/mica2/CC1000Const.h

[22] Jens B. Schmitt and Frank A. Zdarsky, "The DISCO Network Calculator – A Toolbox for Worst Case Analysis", *In Proceedings of the 1st International Conference on Performance Evaluation Methodologies and tools (valuetools'06)*, Pisa, Italy, ACM, November 2006

[23] E. Ilker Oyman and Cem Ersoy, "Multiple Sink Network Design Problem in Large Scale Wireless Sensor Networks", *Proceedings of the IEEE Intern. Conference on Communications*, June 2004, pp. 3663-3667

[24] Haeyong Kim, Yongho Seok, Nakjung Choi, Yanghee Choi and Taekyoung Kwon, "Optimal Multi-sink Positioning and Energy-efficient Routing in Wireless Sensor Networks", *Proceedings of the International Conference On Information Networking (ICOIN)*, 2005, pp. 264-274

[25] Edward O. Wilson, "Sociobiology: The new synthesis", *Belknap Press*, Cambridge, MA 1975

[26] Feature Issue on Particle Swarm Optimization, *International Journal of Computational Intelligence Research 4 (2)*, 2008

[27] Ali R. Guner and Mehmet Sevkli, "A Discrete Particle Swarm Optimization Algorithm for Uncapacitated Facility Location Problem", *Journal of Artificial Evolution and Applications 8 (10)*, January 2008

[28] James Kennedy and Russell C. Eberhart, *Swarm intelligence*, Morgan Kaufmann Publishers Inc., San Francisco, CA, 2001