Coordination of Mobile Agents for Wireless Sensor Network Maintenance

Danny Hermelin, Michael Segal, and Harel Yedidsion

Ben-Gurion University of the Negev, Beer-Sheva, Israel
hermelin, segal, yedidsio @bgu.ac.il

Abstract. In this paper, we study the problem of wireless sensor network (WSN) maintenance using a team of collaborative autonomous mobile agents. The agents are deployed in the area of the WSN in such a way that would minimize the time it takes them to reach a failed sensor and repair it. The team must constantly optimize its collective deployment to account for occupied agents.
The objective is to define the optimal deployment and task allocation strategy, that minimize the solution cost. The solution cost is a linear combination of the weighted sensors' downtime, the agents' traveling distance and penalties incurred due to unrepaired sensors within a certain time limit. A constrained variation of the problem where agents are subject to capacity constraints is also considered. Our solutions are inspired by research in the field of computational geometry and the design of our algorithms is based on state of the art approximation algorithms for the classical problem of facility location.
We empirically compare and analyze the performance of several proposed algorithms. The sensitivity of the algorithms' performance to the following parameters is analyzed: agents to sensors ratio, sensors' sparsity, frequency of failures and repair duration.

1 Introduction

Wireless Sensor Networks (WSN) have recently become a prevalent technology used in a wide range of environmental monitoring applications such as temperature, pollution, radiation and wildlife monitoring. Typically a WSN is composed of a large number of sensor nodes * coupled with short range radio transceivers. The sensors transfer their sensed data to a central hub via multi-hop communication. A communication tree or graph is formed based on physical proximity and must be maintained through technical failures such as battery drainage, battery replacement, solar panel cleaning, memory overload. In case some sensor in the communication tree fails, it not only stops monitoring its environment but, it might also disconnect the communication from other parts of the network.

Practical WSN architecture designs the network with some redundancy so that any node can reach the root in several paths to improve the network's

*We use the terms sensors and nodes interchangeably.

resiliency. Hence, a failure does not necessarily disconnect parts of the network but definitely diminishes the networks robustness.

For more than a decade the use of physical mobile agents (i.e, robots, drowns, autonomous vehicles) has been studied for different aspects of WSN, mainly for data collection and connectivity (See [10, 36] for extensive surveys of such applications). We study the use of mobile agents which have the ability to reach failed sensors, repair them and temporarily replace their role in the task of data collection and transfer. The agents are used to maintain and improve the network's resiliency and reliability.

We aim to optimize the agents' deployment as to minimize the sensors' down-time and the agents' travel distance. We propose algorithms that differ in their positioning methods. The contribution of our work lies in the adaptation of known facility location approximation algorithms for solving the WSN maintenance problem.

2 Problem Definition

Formally, the WSN Maintenance Problem (WMP) is defined as follows: let N be the set of wireless sensor nodes embedded in a two dimensional Euclidean plane with dimensions X and Y. Each sensor node $n_i \in N$ has a weight $w_i \in W$ that reflects its relative importance to the network in terms of data collection and connectivity. Let M be a set of mobile agents that can travel anywhere in the environment and repair sensors that experience technical failures. The sensors are subject to a set of failures F. Each failure ${f_r}^i \in F$ occurs at a certain node (r is the index of the failure and i is the index of the node n_i), at a certain (unknown) start time and requires a *repair-duration* time ${fd_r}^i$, sampled from distribution FD. We assume that the parameters of the distribution are known to the agents. The repair-duration is the time it takes to repair a sensor from the moment an agent has reached it. The response time of a failure ${f_r}^i$, denoted ${rt_r}^i$, refers to the time from when n_i failed until an agent reaches it. Agent m_j's *travel distance* is denoted td_j. The agents' speed is constant and normalized so that they travel one unit of distance in one unit of time. The agents are immediately aware of any failure and can communicate with each other. Once an agent is engaged in repairing a failed node n_i, experiencing failure ${f_r}^i$, it is unable to attend to any other task for the time it takes to travel to n_i plus ${f_r}^i$'s specified repair-duration, ${fd_r}^i$. The agents are limited in their repairing capacity and can only repair a predefined number of sensors, MR per each agent. The number of repairs agent m_j performed is denoted ${m_j}^f$. In addition, if a failure is not repaired within a specified period of time, P_t, the cost increases by an additional penalty ${p_r}^i \in P$ which is a function of the nodes' weight. The total duration of the experiment is denoted E_t. We assume that there is a redundancy in communication paths to the sink in the communication graph so that a single failure does not disconnect parts of the network but it does diminish the network's resiliency and reliability. The **goal** is to find a continuous deployment strategy and a task allocation scheme for the agents, which minimize

the total cost comprised of three objectives; minimize the sum of weighted sensor downtimes, the agents' travel distances, and the penalties. The weighted sensor downtime is a multiplication of the response time and the sensor's weight.

The first part of the cost function, A, is the sum of sensors' weighted downtime, and is formalized as:

$A = \sum_{{f_r}^i \in F}(w_i \cdot {rt_r}^i)$.

The second part of the cost function, B, is the sum of the agents' travel distance:

$B = \sum_{m_j \in M} td_j$.

The third part, C, is the penalty cost calculated as:

$C = \sum_{{f_r}^i \in F} {p_r}^i$.

The total cost is a linear combination of the three parts:

$Cost = \alpha \cdot A + \beta \cdot B + \gamma \cdot C$

The overall objective is to minimize this cost under the agents' repairing capacity constraints. The coefficients α, β and γ are used to assign different proportions to the cost elements. Finally, the WMP framework is defined by a tuple $WMP < N, W, M, F, FD, X, Y, E_t, MR, P_t >$ and the formal objective of the WMP is:

$$\begin{gathered} Given \quad N, W, M, F, FD, X, Y, E_t, MR, P_t \\ minimize\ Cost \\ s.t.\ \forall m_j \in M,\ {m_j}^f \leq MR \end{gathered}$$

2.1 Related Work

Coordinating a team of mobile agents to perform tasks in a dynamic environment is a fundamental problem in AI that has received much attention from researchers [33, 31, 12]. Some of the popular methods that have been used to tackle this problem include: game theory [32, 16], machine learning [3, 37], multi-agent path planning and scheduling [1], distributed constraint optimization [35, 38], economic market based approaches and auctions [19, 22], virtual potential fields [27] and probabilistic swarm behavior [20].

In the field of WSN, mobile elements have been proposed to improve maintenance, data collection, connectivity and energy efficiency [10]. In [30], mobile agents called *mules* (Mobile Ubiquitous LAN Extensions) were first used for data collection in WSN. [8] define the (α, β)-mule problem, where α is the number of simultaneous node failures and β is the number of mobile agents. Unlike our work where the topology of the network is given, their aim is to define the topology of the network in order to minimize the agents' tours. [29] utilizes autonomous mobile base stations (MBSs) to automatically construct new routes to recover disconnected infrastructure, while in [2] mobile backbone nodes (MBNs) are controlled in order to maintain network connectivity while minimizing the number of MBNs that are actually deployed. The paper by [34] addresses the problem of restoring connectivity in a WSN, using a mobile agent to place relay and sink nodes.

The use of facility location techniques was proposed for the coordination of mobile agents for WSN maintenance by [14]. However, that work only considered uniform sensor weights, deterministic repair durations, and infinite repairing capacity. We aim to further contribute to the study of multi-agent coordination by investigating the integration of modified facility location techniques into the design of the algorithms which are meant to solve the WSN maintenance problem. We continue to develop this line of research by considering more realistic assumptions such as non uniform sensor weights, limited agents' repairing capacity, and non-deterministic repair durations. The WMP has many similarities with the k-Server problem proposed by [21] In this problem, a set of k servers must reach demand points, one server per demand, and demands arrive on-line. However, the major difference is that the objective of the k-Server problem is to minimize the total traveling distance of the servers while in our problem this is a secondary objective. Our main interest is finding an optimal deployment scheme for the agents as to minimize response times. The k-Server problem does not introduce the notion of repair-duration, and it is sufficient for a server to simply reach a demand point to satisfy it. Contrarily in our problem, the server must wait and service the sensor for the repair-duration period. This difference has a considerable influence on the design of efficient algorithms. In the k-Server problem a task allocation strategy which sends the closest server might work well as proposed by [21], yet in our case, it may cause large response times. One practical application of the k-Server problem is the Emergency Medical Services (EMS) problem [5]. However, the key difference from our model is that in EMS the set of deployment bases is finite and pre-defined [15], where in our case the servers can be located anywhere in the plane. The static version of a single iteration of our problem relates to the Facility Location problem for which there are two main variants; the k-Center and k-Median problems (see Definitions 3 and 1 respectively). Therefore, we use known solution method for these problems in the design of our algorithms. Since both of these problems are NP-Hard, we integrate into our algorithms modified versions of known greedy heuristics for these problems that guarantee some approximation ratio of the optimal solution. These solutions run in polynomial time and thus are suited for practical applications that require quick responses. Specifically, Gonzalez [13] proposed the Farthest-First (FF) algorithm that provides a 2-approximation for the k-Center problem in $O(n)$ time. In [7] it is shown that the reverse greedy algorithm (RGreedy) guarantees at most an approximation ratio of $O(\log n)$ in $O(n^2 \log n)$ time for the k-Median problem.

3 Algorithms

In this section we define the algorithms for solving the WMP. Due to the similarity of our problem to the Facility Location problem, we integrate known solution techniques to this problem into our algorithms.

3.1 Facility Location Strategies

The Facility Location problem is a well studied problem in computer science and operations research [28]. It has many variations which primarily deal with optimally placing k facilities to serve n given cities. Two classical variants of this problem which are closely related to our problem are the weighted k-Median and weighted k-Center problems, both proven NP-hard problems. [17, 18]. Consequently, there is a significant body of work that deals with approximation algorithms for these problems. In what follows we describe these problems, the heuristics proposed to solve them, and the modifications we made to adjust them to solve the WMP.

Definition 1 *The **Weighted k-Median** problem is defined as follows. Given a set of n points, N, each point n_i having a weight w_i, and an integer k, find a set M of k points for which the sum of weighted distances, i.e., the weight of any point in N, multiplied by the euclidean distance to its closest point in M, is minimum.*

An optimal solution to the weighted k-Median problem minimizes the weighted sum of distances of all the cities from their closest facility and thus, any algorithm that provides a good solution for this problem can be useful in the design of an algorithm that would minimize average response times. The Reverse Greedy algorithm (RGreedy) proposed in [7] to solve the weighted k-Median problem, works as follows. It starts by placing facilities on all nodes. At each step, it removes a facility to minimize the total weighted distance to the remaining facilities. It stops when k facilities remain. It runs in $O(n^2 \log n)$ time and guarantees a $\log n$ approximation of the optimal weighted k-Median solution.

Definition 2 *Let N be a set of points, each point n_i having weight w_i, and coordinates r_i. The **Weighted Centroid** of N, $wc(N)$, also called the center of mass of N, is the weighted arithmetic mean of their coordinates. $wc(N) = \frac{1}{\sum w_i} \sum_{n_i \in N} w_i r_i$.*

In [23] it was proven that a centroid of a set of points N provides a 2-approximation for the 1-Median of N. We extend this result to the weighted case in Theorem 1. Since it is not a straightforward to assume that the approximation ratio holds in the weighted case, we also provide the necessary proof for this novel result.

Theorem 1. *For any set of weighted points N with a weighted centroid $wc(N)$, and an arbitrary point p in the Euclidean plane,*
$\sum_{n_i \in N} d(n_i, wc(N)) w_i \leq 2 \sum_{n_i \in N} d(n_i, p) w_i$.

Proof. Let $p*$ be the optimal weighted median of N. From the triangle inequality we get:

$$\forall n_i \in N, d(n_i, wc(N)) \leq d(n_i, p*) + d(p*, wc(N)).$$

By multiplying by the weights we obtain:

$$\forall n_i \in N, d(N_i, wc(N))w_i \leq d(n_i, p*)w_i + d(p*, wc(N))w_i.$$

After summing over all the points in N we get:
$\sum_{n_i \in N} d(n_i, wc(N))w_i \leq \sum_{n_i \in N} d(n_i, p*)w_i + d(p*, wc(N)) \sum_{n_i \in N} w_i$. From the convexity of the Euclidean norm it follows that the norm of a weighted average of a set of points is at most the average of the weighted norms of the points in the set, that is: $d(p*, wc(N)) \leq \sum_{n_i \in N} d(p*, n_i)w_i / \sum_{n_i \in N} w_i$. Therefore, for any $p \in R^2$,
$\sum_{n_i \in N} d(n_i, wc(N)) \leq 2 \sum_{n_i \in N} d(n_i, p*) \leq 2 \sum_{n_i \in N} d(n_i, p)$

Definition 3 *The **Weighted** k-**Center** problem is defined as follows. Given a set of n points, N, each point n_i having weight w_i, and an integer k, find a set M of k points for which the largest weighted distance, i.e., the weight of any point in N, multiplied by the euclidean distance to its closest point in M is minimum.*

[9] proposed an algorithm that provides a $min(3, 1 + \rho)$ approximation for the weighted k-Center problem in $O(kn)$ time, where ρ is the ratio between the maximal and minimal weights. The algorithm selects k points in the following way. The first point is positioned on the node with the maximal weight. The algorithm continues to choose the next point out of the remaining node locations, on the node whose minimal weighted distance from the set of previously selected points is the largest. It stops when k point are chosen. This algorithm was not named in [9]. However, it represents a modified version of the Farthest-First algorithm proposed in [13] that provides a 2-approximation for the unweighted k-Center problem. Hence, we will refer to it as the Weighted Farthest First (WFF) algorithm.

It should be mentioned that there are additional approaches to deal with variants of k-Center and k-Median problems. So-called ε-nets [25] and Linear Programming relaxation [6] are just a few examples. However, these approaches do not allow distributed implementation which is essential to make our solutions feasible for real life deployments.

3.2 Proposed Algorithms

We propose three algorithms based on the facility location approximation algorithms, which differ in their redeployment strategies. We also present two other algorithms to benchmark against and compare the performance of our algorithms. The task allocation method we use for all the algorithms is the *Closest-Available* allocation, i.e, send the agent that would arrive the soonest to a failure, while taking into account the estimated remaining repair time. Preliminary tests not shown here have indicated that this task allocation strategy is more efficient than other strategies such as the Closest or the Closest-Least-Traveled. Here are the proposed algorithms:

No-Redeployment Algorithm (NRD) This algorithm is designed to be a baseline algorithm to which we will compare the others. In our preliminary testing it provided the best results out of the algorithms that do not perform any redeployment. It uses uniform grid placement for the initial agents' deployment. The closest-available agent is sent to repair each failure and no redeployment is performed by the other agents.

Weighted k-Center Algorithm (*W*k-Center) This algorithm uses WFF for the initial agents' deployment and for their redeployment. Redeployment is performed following every failure. The available agents are redeployed according to the WFF algorithm and the occupied agents are disregarded. After recalculating the new positions, we are faced with a classical Assignment Problem, i.e., how to assign the agents to new locations while minimizing the total traveling distance during redeployment. The Hungarian Algorithm, [24] is a combinatorial optimization algorithm that solves the assignment problem in polynomial time. It runs in polynomial time with regards to the number of assignments i.e., $O((k')^3)$ time, where k' is the number of available agents. In [11] this runtime was improved for the Multi-Robot Task Allocation Problem and the algorithm comes up with a global optimum solution in $O((k')^3)$ cumulative time, but with $O((k')^2)$ for each robot, with $O((k')^3)$ number of messages exchanged among the n robots. This improvement is important for our implementation since the $Wk-Median$ algorithms runs in $O((k')^2 \log(k')$ and we do not want the assignment problem solution to be the bottleneck of the computation process.

Weighted k-Median Algorithm (*W*k-Median) This algorithm uses RGreedy for the initial agents' deployment and for their redeployment. During redeployment, only $(k-b)$ medians are calculated out of $(n-b)$ node locations where b is the number of busy, occupied agents. Occupied agents and nodes that are being repaired are disregarded in the RGreedy calculation. Here too, the Hungarian Algorithm is used to optimally assign agents to new locations.

Weighted k-Centroid Algorithm (*W*k-Centroid) The initial deployment is done using RGreedy but is immediately followed by an iterative procedure called weighted centroid-adjustment, where each agent moves to the weighted centroid of the nodes in its Voronoi cell. Weighted centroid-adjustment is also used to redeploy the agents after every movement.

Local Search Algorithm While approximation algorithms can provide certain theoretical guarantees, there is no guarantee that they perform as well as local search methods in practice. For this reason we also implemented a local search algorithm, based on the scheme proposed in [4], to evaluate the overall performance of our proposed algorithms which are based on approximation algorithms. The agents' deployment strives to achieve the k-Median criterion. The search process is performed in a distributed concurrent manner. At each

iteration, every agent moves to a nearby position that minimizes the sum of distances from it to the nodes closest to it. The number of iterations is limited by either a constant, which equals the number of nodes n, or by convergence, thus maintaining $O(n)$ runtime. Here too, the closest-available agent is sent to any failure.

4 Experimental Evaluation

In order to compare the performance of the different algorithms, we developed a software simulator, representing the WMP. The area of the simulated problem is an X over Y plane. Any number of nodes (N), and agents (M) can be positioned in the area. Nodes have weights (W) associated with their importance. Within the total duration of the experiment (E_t), we can induce any number of failures (F) on the nodes. The nodes which fail are chosen uniformly at random from N and the start time of each failure is chosen uniformly at random from $(0, E_t)$. The values of the *repair-durations* are drawn from a Lognormal (LN) distribution which is commonly used to model repair times of a maintained system [26]. It is a better fit than the Normal distribution since service times cannot be negative. In each experiment the specific values for $N, W, M, E_t, F, FD, X, Y, MR, and\ P_t$ are chosen differently to demonstrate the algorithms' behavior in different scenarios. We performed rigorous experimental evaluations to analyze the sensitivity of the algorithms' performance to various parameter values. Each reported result represents an average of 50 random experiments. In each experiment the initial agents' locations, nodes' locations, failure locations and start times are randomly selected. We used the same set of random seeds so that each algorithm is presented with the same 50 randomly generated problems. To analyze the statistical significance of the results, we performed T-Tests to validate the differences between the algorithms' performances. We state verbally in the text whether the difference between the results are statistically significant (i.e. p-value < 0.05) or not. For sake of readability, we refrain from adding error bars to the figures.

In the following subsections we analyze the sensitivity of the algorithms' performance to different parameter settings and modeling constraints. If not stated otherwise, the default experimental setting include problems with $|M|$=10 agents, $|N|$=100 nodes that are randomly deployed in a $X = 100$ over $Y = 100$ area. $|F|$=50 failures are generated with repair-duration$\sim LN(100, 10)$. Failure distribution is uniform i.e, each sensor has the same probability of failing. Node weights W are normalized to 1. In our setup minimizing the down-time of the WSN is a primary objective compared to keeping a healthy maintenance team by minimizing the agent's movement and that is reflected in the chosen coefficient values which are $\alpha = 5, \beta = 1, \gamma = 1$. The experiment duration is $E_t = 10000$.

4.1 Ratio of agents to sensors

In this subsection we analyze the algorithms' sensitivity to the ratio of agents to sensors. In this experiment we do not impose capacity constraints or penalties i.e., $\gamma = 0, MR = Inf$.

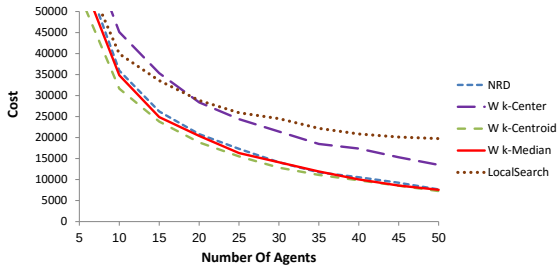

Fig. 1. Sensitivity to the ratio of agents to sensors.

Figure 1 presents a sensitivity analysis to the ratio of agents to sensors. The number of nodes is 100 and the number of agents range from 5 to 50 which corresponds to agent-to-node ratios of 1:20-1:2 respectively. The total cost is compared between the algorithms for each ratio. Not surprisingly, all algorithms produce lower cost solutions as more agents are available. It is evident that the NRD, Wk-Median and Wk-Centroid algorithms produce similarly good results with respect to Wk-Center and Local Search in all ratios. The difference between the results of Wk-Median and Wk-Centroid to those of Wk-Center and Local Search are statistically significant.

4.2 Frequency of failures

This subsection presents a comparison between the algorithms in varying frequencies of failures. The experimental setting used 0,5,10,...,50 failures. In this experiment we do not impose capacity constraints or penalties i.e., $\gamma = 0, MR = Inf$.

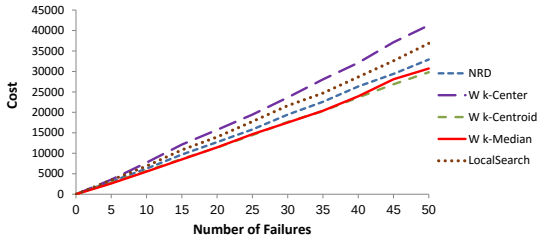

Fig. 2. Cost for the number of failures in a given time frame.

The results depicted in Figure 2 demonstrate the advantage of the Wk-Centroid algorithm in terms of minimizing the total cost. It is clear that as there are more failures in a given time frame, the solution cost of all the algorithms grows. The Wk-Centroid produced the best results in all tested number of failures values. The difference between the results of Wk-Centroid to those of Wk-Center, Local Search and NRD are statistically significant.

4.3 Repair-duration

In this subsection we measure the cost as a function of different mean and standard deviation of repair-durations with distribution $\sim LN(100, 10) - LN(1000, 100)$. In this experiment we do not impose capacity constraints or penalties i.e., $\gamma = 0, MR = Inf$. Figure 3 presents a comparison between the algorithms when

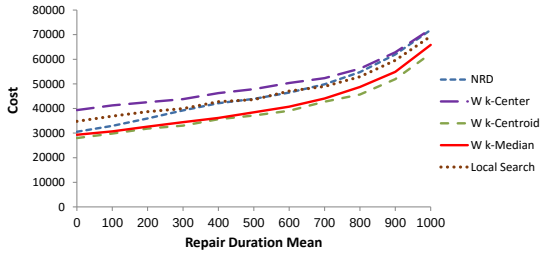

Fig. 3. Cost as a factor of increasing repair-durations.

failures require different repair-durations. This parameter is important since it justifies the need for redeployment. If repair-durations are large in comparison with the time between failures, the chance of a failure occurring in the vicinity of an occupied agent increases. The larger the repair-durations, the worse the cost becomes for all algorithms. However, it is interesting to see that the NRD performs steadily worse in comparison to the other redeploying algorithms as repair-durations increase. The Wk-Centroid algorithm achieves the lowest cost in all durations. Both Wk-Centroid and Wk-Median perform significantly better than Wk-Center.

4.4 Sensors' sparsity

In this subsection we analyze different environment sizes to check how the node density influences the algorithms' performance. In this experiment we do not impose capacity constraints or penalties i.e., $\gamma = 0, MR = Inf$. Figure 4 presents a

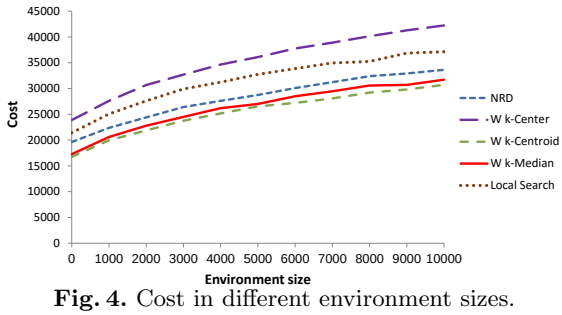

Fig. 4. Cost in different environment sizes.

comparison between the algorithms in different environment sizes. As the environment grows, the average traveled distance increases and so does the average downtimes and as a result, the average cost for all algorithms. The Wk-Centroid algorithm produces the best results. The difference between the results of Wk-Median and Wk-Centroid to those of Wk-Center and Local Search are statistically significant.

4.5 Agents' repairing capacity

In this subsection we introduce the notion of limited repairing capacity for the agents. This constraint is more realistic as agents may have a limited number of replacement parts for repairing the sensors. When agents reach their maximal capacity they become inactive. It is assumed that better load balancing would lead to less inactive agents and as a result achieve lower costs. In this experiment we do not impose penalties. Figure 5 presents a comparison between the

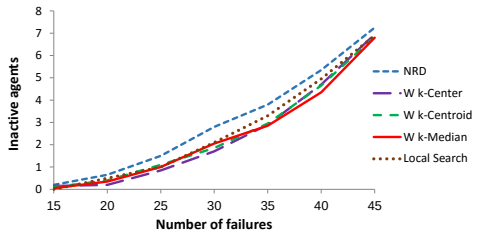

Fig. 5. Num. of inactive agents for increasing number of failures.

algorithms in terms of the average number of agents that became inactive after they reached their maximal repairing capacity. The number of failures is 0-50, there are 10 agents each with a capacity to repair 5 failures. Clearly, as more failures are introduced, the agent team as a whole reaches its maximal repairing capacity. However, it is evident that NRD has a significantly higher number of inactive agents than the redeployment algorithms. The Wk-Center has inherent load balancing properties as it aims to minimize the maximal distance between sensors and their closest agent. This characteristic results in a low number of inactive agents. Figure 6 shows that the total cost of all the algorithms increases compared to the non constrained version (See Figure 2). However, the inherent load balancing by the Wk-Center algorithm indeed results in a better performance compared to NRD and Local Search, and in the capacitated scenario it is no longer the worst performing algorithm.

4.6 Penalties

In previous subsections, agents were not penalized for long response times. In this subsection we introduce the notion of penalties. Controlling the maximum

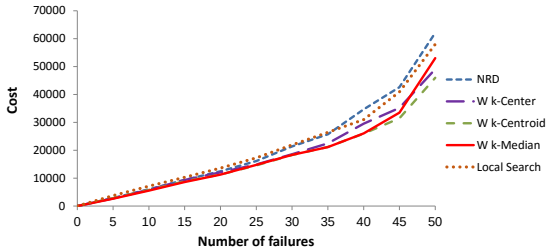

Fig. 6. Cost under capacity constraints

downtime is important in WSN where communication cycles happen every fixed period of time. Limiting the maximal downtime ensures that no more than a known number of communication cycles are missed.

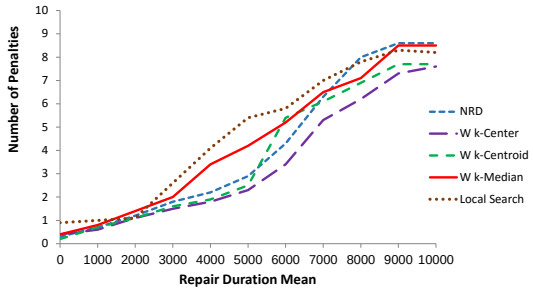

Fig. 7. Num. of penalties for increasing repair-durations.

Figure 7 presents a comparison of algorithms in terms of the number of penalties incurred. The results show that all algorithms incur more penalties as repair-duration increases, yet the Wk-Center algorithm, which is designed to minimize the maximal response time, is the least sensitive algorithm. In this experiment repair-durations means range from 0-10000 to simulate extremely long delays which cause more penalties.

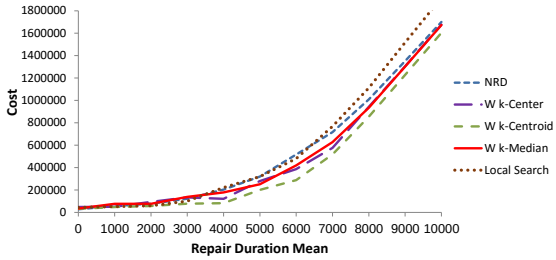

Fig. 8. Cost under penalty constraints

Figure 8 shows how less penalties incurred translate to lower costs and the Wk-Center now produces similar results to those of Wk-Median and Wk-Centroid. The Wk-Centroid bears the lowest overall costs but the differences between the results in this case were not statistically significant.

4.7 Runtime Analysis

In this section we analyze the actual execution time it takes to run each algorithm. The experiments were run on two different machines with the following characteristics:

1. A Dell Latitude laptop with a 64 bit Windows10 OS, an Intel Core i7 CPU running at 2.7 GHz with 8GB RAM.
2. A server platform with a 64 bit Windows Server 2008 OS, two Intel Xeon precessors running at 2.4 GHz with 20GB RAM.

We ran 1000 experiments on each machine, using various parameter settings. The average time over all experiments and machines to compute a redeployment by a node for each algorithm in milliseconds showed that the NRD algorithm is the fastest with 0.3ms followed by Wk-Center with 0.8ms. The next is the Wk-Centroid with 2.7ms and the Local Search with 3.7ms. Finally, the Wk-Median is the slowest algorithm with 12.2ms. While the Wk-Median algorithm performs significantly slower than the rest of the algorithms, the millisecond differences in run-times are negligible in the context of mobile robotics.

5 Conclusion

In this paper we define and study the WSN maintenance problem (WMP), where a team of mobile agents are charged with maintaining a WSN. In WMP, failures have non deterministic repair-durations and sensor nodes have weights that indicate their importance in the communication network and in data collection. The goal is to design efficient algorithms that minimize a unified cost function that incorporates the sensors' weighted downtime and the agents' travel distance.

Our proposed algorithms are inspired by facility location approximation algorithms for solving the WMP. The algorithms are all computationally tractable to enable each agent to compute the redeployment and node allocation efficiently after every movement, in a distributed manner without any need for central control. Specifically, we evaluate the performance of three algorithms based on approximation algorithms for the Wk-Center and Wk-Median problems and compare them to two baseline algorithms. The first, NRD, is the best performing algorithm which does not use any redeployment techniques. The second, Local Search, enables to compare the algorithms to a heuristic local search approach.

Our empirical results indicate that the Wk-Centroid approach outperforms other algorithms in a wide range of settings. Its performance is closely matched by that of the Wk-Median algorithm, both designed to minimize the average

response time. The Wk-Center algorithm provides the worst results in most settings except for the number of penalties given due to late response times, where it incurred the least number of penalties.

Finally, our contribution in this paper is the continued investigation of the use of facility location and computational geometry techniques into the design of algorithms used to coordinate a team of mobile agents in performing complex tasks such as WSN maintenance.

References

1. Agmon, N., Urielli, D., Stone, P.: Multiagent patrol generalized to complex environmental conditions. In: AAAI (2011)
2. Anand, S., Zusseman, G., Modiano, E.: Construction and maintenance of wireless mobile backbone networks. IEEE/ACM Transactions on Networking **17.1**, 239–252 (2009)
3. Andreas, K., Ajit, S., Carlos, G.: Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. Journal of Machine Learning Research **9**, 235–284 (2008)
4. Arya, V., Garg, N., Khandekar, R., Meyerson, A., Munagala, K., Pandit, V.: Local search heuristics for k-median and facility location problems. SIAM Journal on computing **33**(3), 544–562 (2004)
5. Brotcorne, L., Laporte, G., Semet, F.: Ambulance location and relocation models. European journal of operational research **147**(3), 451–463 (2003)
6. Charikar, M., Guha, S., Tardos, É., Shmoys, D.B.: A constant-factor approximation algorithm for the k-median problem. J. Comput. Syst. Sci. **65**(1), 129–149 (2002)
7. Chrobak, M., Kenyon, C., Young, N.E.: The reverse greedy algorithm for the metric k-median problem. Computing and Combinatorics Conference pp. 654–660 (2005)
8. Crowcroft, J., Levin, L., Segal, M.: Using data mules for sensor network data recovery. Ad Hoc Networks **40**, 26–36 (2016)
9. Dyer, M.E., Frieze, A.M.: A simple heuristic for the p-centre problem. Operations Research Letters **3**(6), 285–288 (1985)
10. Francesco, M.D., Das, S.K., Giuseppe, A.: Data collection in wireless sensor networks with mobile elements: A survey. ACM Transactions on Sensor Networks (TOSN) **8.1**, 7–38 (2011)
11. Giordani, S., Lujak, M., Martinelli, F.: A distributed algorithm for the multi-robot task allocation problem. In: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems. pp. 721–730. Springer (2010)
12. Goldman, C.V., Zilberstein, S.: Decentralized control of cooperative systems: Categorization and complexity analysis (2004)
13. Gonzalez, T.F.: Clustering to minimize the maximum intercluster distance. Theoretical Computer Science **38**, 293–306 (1985)
14. Hermelin, D., Segal, M., Yedidsion, H.: Coordination of mobile mules via facility location strategies. In: International Conference on Practical Applications of Agents and Multi-Agent Systems. pp. 107–119. Springer (2017)
15. Jagtenberg, C., Bhulai, S., van der Mei, R.: An efficient heuristic for real-time ambulance redeployment. Operations Research for Health Care **4**, 27–35 (2015)
16. Jiang, A., Procaccia, A., Qian, Y., Shah, N., Tambe, M.: Defender (mis) coordination in security games. In: AAAI (2013)

17. Kariv, O., Hakimi, S.: An algorithmic approach to network location problems. i: The p-centers. SIAM Journal on Applied Mathematics **37(3)**, 513–538 (1979)
18. Kariv, O., Hakimi, S.: An algorithmic approach to network location problems. ii: The p-medians. SIAM Journal on Applied Mathematics **37(3)**, 539–560 (1979)
19. Koenig, S., Keskinocak, P., Tovey, C.: Progress on agent coordination with cooperative auctions (2010)
20. Konur, S., Dixon, C., Fisher, M.: Analysing robot swarm behaviour via probabilistic model checking. Robotics and Autonomous Systems **60(2)**, 199–213 (2012)
21. Manasse, M.S., McGeoch, L.A., Sleator, D.D.: Competitive algorithms for server problems. Journal of Algorithms **11**(2), 208–230 (1990)
22. McIntire, M., Nunes, E., Gini, M.: Iterated multi-robot auctions for precedence-constrained task scheduling. In: AAMAS (2016)
23. Milyeykovski, V., Segal, M., Katz, V.: central nodes for efficient data collection in wireless sensor networks. Computer Networks **91**, 425–437 (2015)
24. Munkres, J.: Algorithms for the assignment and transportation problems. Journal of the society for industrial and applied mathematics **5**(1), 32–38 (1957)
25. Mustafa, N.H., Ray, S.: Improved results on geometric hitting set problems. Discrete & Computational Geometry **44**(4), 883–895 (2010)
26. O'connor, P.D., O'Connor, P., Kleyner, A.: Practical reliability engineering. John Wiley & Sons (2012)
27. Poduri, S., Sukhatme, G.S.: Constrained coverage for mobile sensor networks. In: ICRA. pp. 165–171 (2004)
28. ReVelle, C., Eiselt, H.: Location analysis: A synthesis and survey. European Journal of Operational Research **165(1)**, 1–19 (2005)
29. Rui, T., Li, H., Miura, R.: Dynamic recovery of wireless multi-hop infrastructure with the autonomous mobile base station. IEEE Access **4**, 627–638 (2016)
30. Shah, R.C., Roy, S., Jain, S., Brunette, W.: Data mules: Modeling and analysis of a three-tier architecture for sparse sensor networks. Ad Hoc Networks **1(2)**, 215–233 (2003)
31. Stone, P., Kaminka, G., Kraus, S., Rosenschein, J.: Ad hoc autonomous agent teams: Collaboration without pre-coordination (2010)
32. Tambe, M.: Security and game theory: algorithms, deployed systems, lessons learned. Cambridge University Press (2011)
33. Tambe, M., Shen, W., Mataric, M., Pynadath, D., Goldberg, D., ans Z. Qiu, P.M., Salemi, B.: Teamwork in cyberspace: Using teamcore to make agents team-ready (1999)
34. Truong, T.T., Brown, K.N., Sreenan, C.J.: Repairing wireless sensor network connectivity with mobility and hop-count constraints. In: International Conference on Ad-Hoc Networks and Wireless. pp. 75–86. Springer (2013)
35. Yeoh, W., Felner, A., Koenig, S.: Bnb-adopt: An asynchronous branch-and-bound dcop algorithm (2008)
36. Younis, M., Senturk, I.F., Akkaya, K., Lee, S., Senel, F.: Topology management techniques for tolerating node failures in wireless sensor networks: A survey. Computer Networks **58**, 254–283 (2014)
37. Zhang, C., Lesser, V.R.: Coordinated multi-agent reinforcement learning in networked distributed pomdps. In: AAAI (2011)
38. Zivan, R., Yedidsion, H., Okamoto, S., Glinton, R., Sycara, K.P.: Distributed constraint optimization for teams of mobile sensing agents. Journal of Autonomous Agents and Multi-Agent Systems **29**, 495–536 (2015)