# Maneuverable Relays to Improve Energy Efficiency in Sensor Networks

Stephan Eidenbenz, Lukas Kroc, James P. Smith
CCS-5, MS M997; Los Alamos National Laboratory; Los Alamos, NM 87545.
Email: {eidenben, kroc, jpsmith}@lanl.gov

## Abstract

*We propose a hybrid model to alleviate the notorious problem of premature battery depletion in sensor networks: in a first stage, simple sensors are deployed over an area of interest using any traditional method, in a second phase, more powerful relays are put in positions that allow them to off-load as much of the forwarding burden imposed on the simple sensors as possible. We introduce this model in detail, give a few theoretical results with respect to the optimal placement of the powerful sensors, propose various heuristic approaches for their placement, and present comparative simulation results for these heuristics.*

## 1. Introduction

It is well established that while sensor networks promise unprecedented opportunities for data collection in a great variety of scenarios, their large-scale deployment is hindered by their thirst for energy. Considerable research efforts have gone into developing energy-efficient protocols on all stack layers for the sensor network setting in which communication to a single base station happens in a multi-hop fashion with intermediate transceiver sensors being asked to forward data. While the multiple hop scheme is certainly an improvement in terms of energy efficiency over older schemes that require each sensor to emit a signal strong enough to reach the base station in a single hop, it still falls short of real-life energy efficiency requirements in many applications, not to mention that the reliability of such schemes usually decreases drastically as routing paths get longer.

We propose a hybrid model that has the potential to reduce energy requirements—a claim that we support by simulation evidence in the last part of this paper. In our model, there are two different types of transceivers:

- **Sensors** are equipped with a sensor to measure some environmental variable (such as temperature or radiation levels) and with a low-cost, short-range transceiver that is able to receive data packets from other sensors and to send data packets that it either generated itself or received from other sensors. Moreover, each sensor is equipped with a GPS-like device allowing it to determine its position relative to some reference frame.
- **Relays** are more powerful transceivers that are not necessarily equipped with a sensor. They receive data packets from sensors and forward them directly to a fixed-infrastructure base station using their medium-range transmission capabilities. Relays are also equipped with a GPS-like capability.[1]

In the application scenario that we envision, the sensors are deployed in a first stage, the relays are deployed in a second stage, and actual data traffic starts being transmitted only in a third phase. A key element of our model is that relays are maneuverable: they can be deployed to locations specified in coordinates; this can be achieved in several ways depending on the application (e.g., by moving them manually to their positions or by equipping them with a motor and letting them drive to their positions). Figure 1 illustrates the hierarchical architecture of the sensor network.

For ease of illustration, we assume a unit disk graph model (i.e., each sensor has the same fixed transmission range) despite its deficiencies (generalizations to more involved physical models are feasible). In order to avoid the complication of collisions, we assume a TDMA model on the MAC layer with each transceiver being assigned a unique time-slot.[2] A more precise meta-algorithm for the

1 Additional energy savings may be possible by replacing standard GPS devices by a scheme where a small number (at least three) relays are placed on the boundary of the area in which the sensors are distributed. The relays can then send out signals that are received by the sensors; based on the signal strengths from the different relays, the sensors can then compute their position (similar to the way real GPS works). To eliminate the necessity for sensors to be able to detect signal strength, the relays can also send out consecutive messages of increasing signal strength; based on the first intercepted message from each relay, a sensor can compute its location fairly accurately.

2 Since not all sensors are within each others' transmission range, TDMA requires global time synchronization. Fortunately, a global clock is a feature of any GPS-like system, thus time synchronization

Proceedings of the 3rd Int'l Conf. on Pervasive Computing and Communications Workshops (PerCom 2005 Workshops)
0-7695-2300-5/05 $20.00 © 2005 IEEE

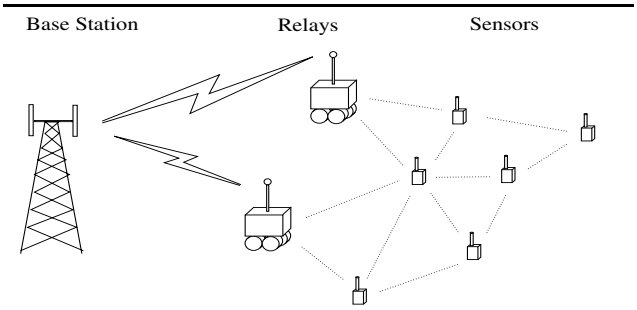


**Figure 1. Schematic illustration of the hierarchical levels of the sensor network**

scenario is as follows:

1. Sensors are deployed; each computes its own position.
2. A single relay is placed into the center of the sensor spread area.
3. The relay sends out a beacon packet indicating that it is the relay by setting the hop-count field in the packet to zero. Each sensor receives such a packet, stores the ID of the sender of the packet, increases the hop count by one and forwards the packet. At the end of this flooding phase, each sensor knows its hop-count distance to the relay and also knows to which neighbor it needs to forward a packet when sending to the relay.
4. Each sensor informs the fixed-infrastructure base station about its position by sending a packet with position information along the computed shortest path through the relay.
5. From the sensor positions, the base station computes the relay positions using an algorithm or heuristic to find a good solution.
6. The relays are programmed with their final coordinates; they then move or are moved to their positions.
7. Each relay sends out a beacon packet that is propagated by the sensors as in Step 3. Each sensor may receive several of these packets from different relays or intermediary sensors, and chooses randomly among those on the shortest paths.
8. Data traffic starts with data packets being forwarded along the computed shortest paths to a relay who then sends the data to the fixed base station.

The computational work that the base station needs to carry out in order to compute the target positions of the relays in Step 5 is key to the overall energy use of the system. As is common, we assume that the base station has unlimited transmission power and is equipped with a powerful computer. Let $S = \{1, \ldots, n\}$ denote the IDs of the $n$ sensors; let the hop-count distance between a sensor node $i$ and its associated relay (i.e., its hop-count closest relay) be $d_i$; finally, let $f_i$ (for $1 \leq i \leq n$) denote the average number of data packets that sensor $i$ generates in a time unit, and let $E_U$ denote the amount of energy that a sensor needs to spend to forward one data packet. With this notation, the amount of energy $E_S$ used by the sensors in the system in a time unit (power) is

$$E_S = \sum_{i=0}^{n} f_i \cdot d_i \cdot E_U, \tag{1}$$

since each data packet that a sensor node $i$ sends out needs to travel $d_i$ hops before it reaches its relay. Thus, we can formulate a formal optimization problem RELAY POSITIONING: Given $n$ positions of sensors in the plane and an integer $k$ (and for a decision version, an energy threshold $E_T$), find $k$ relay positions such that the resulting energy usage $E_S$ is minimum. Alternatively, for the decision version, find $k$ relay positions such that the resulting energy usage $E_S$ is at most $E_T$.

We will focus on this aspect of the proposed sensor network, as other aspects are fairly straight-forward. We study RELAY POSITIONING from both a theoretical point of view and from an applied point of view through simulation. We present theoretical results in Section 2, a few heuristic approaches for solving the problem in Section 3, and simulation results in Section 4. Pointers to related work and a brief conclusion are given in Section 5.

## 2. Theoretical Results

In this section, we first show that in order to find optimum solutions for RELAY POSITIONING, we only need to consider a discrete set of positions in the plane where relays can be placed. We then present a proof that our RELAY POSITIONING is in fact $NP$-hard (despite its special structure), thus forcing us to resort to approximation algorithms or heuristics to find good solutions.

**Discretization.** We are assuming a unit disk graph model among a finite set of sensor nodes, so the set of sensor nodes from which a relay can receive packets changes only along a finite set of arcs on the plane. Assume we are given the positions of the $n$ sensor nodes. If we draw a unit-radius circle around each sensor, we obtain a partitioning of the plane into cells whose boundaries are circle segments. The set of sensors that can reach a relay does not change within a cell, so it does not matter where in the interior of a cell we place a relay. A relay on the boundary of the cell is within reach of the same set of sensors as in the interior plus possible additional sensors. By definition of the unit disk graph, the set of sensors within reach of a relay changes only along cell

is not much of an additional burden.

boundaries, including the intersection points of each pair of circles. When searching for optimum relay positions, we can limit ourselves to circle intersection points because all points on each cell defining boundary are equivalent in this sense, and those two points are guaranteed to represent the more connected cells. Since each circle can intersect each other circle at most twice, the total number of intersection points is $O(n^2)$. Thus, we have:

**Lemma 1** *For any given instance of* RELAY POSITIONING*, there exists an optimum solution in which each relay is positioned at one of the $O(n^2)$ unit circle intersection points.*

With this discrete characterization, we can formulate our problem as a special case of MAXIMUM $k$-FACILITY LOCATION [5] with the node set consisting of all sensor positions and circle intersection points, profits between sensor nodes and potential relay positions defined as the negative of the shortest hop count distances, other profits set to zero. A greedy heuristic achieves an approximation ratio of $\frac{e}{e-1}$ for MAXIMUM $k$-FACILITY LOCATION and therefore also for RELAY POSITIONING. However, the fact that MAXIMUM $k$-FACILITY LOCATION is $NP$-hard does not imply that RELAY POSITIONING is $NP$-hard since the latter is a special case of the former.

**Hardness.** We now show that the decision version of RELAY POSITIONING is $NP$-complete by proposing a polynomial-time reduction from DOMINATING SET ON PLANAR GRAPHS WITH MAXIMUM DEGREE 4, which is known to be $NP$-hard [6]. This reduction is inspired by a reduction for a similar problem of [2]. We will also need a graph drawing result:

**Lemma 2** *[7] A planar graph with maximum degree 4 can be embedded in the plane using $O(|V|)$ area such that its vertices $V$ are at integer coordinates and its edges consist of (possible multiple) horizontal or vertical line segments each of which has end points with integer coordinates.*

We are now ready to prove our results:

**Theorem 1** *The decision version of* RELAY POSITIONING *is $NP$-complete.*

*Proof:* Lemma 1 shows that RELAY POSITIONING in its decision version is in $NP$: even an optimum solution can be described as a set of positions each of which is an intersection point of two circles around input data points. Thus, a solution is of polynomial length in the input, and its validity can be verified in polynomial time.[3]

3 Proving $NP$-membership is not as trivial for this geometric problem as it is for most combinatorial problems. We need to explicitly show that we can express the coordinates of solution positions with a polynomial number of bits.

To show that RELAY POSITIONING is $NP$-hard, we transform an instance $I$ of dominating set on planar graphs with maximum degree 4. Instance $I$ consists of a graph $G(V, E)$ with $n$ nodes $V = \{v_1, \ldots, v_n\}$ and edges $E \subseteq V^2$, and an integer $k$ (for answering the question whether a dominating set of at most $k$ nodes exists). We assume that we are given the input graph embedded in the plane according to Lemma 2 with node $v_i$ at integer coordinates $(x_i, y_i)$. Let $l_{i,j}$ denote the Euclidean length of the horizontal and vertical line segments that form edge $(v_i, v_j)$ in the embedding. By Lemma 2, $l_{i,j}$ is of polynomial length.

We transform instance $I = (G, k)$ into a RELAY POSITIONING instance $I' = (S, k', E_T)$ that consists of a total of $n' = n + \sum_{(i,j)\in E}(12l_{i,j} - 3)$ sensor nodes $S$ that are spread along the edge lines in the embedding of $I$ as follows: a sensor node $s_i$ is placed at the embedded position of node $v_i$; on the line segment of length $l_{i,j}$ representing edge $(i, j)$, an additional $12l_{i,j} - 3$ sensors $S^{i,j} = \{s_1^{i,j}, \ldots, s_{12l_{i,j}-3}^{i,j}\}$ are placed consecutively at certain distances. More precisely, sensor $s_c^{i,j}$ for $1 \le c \le 5$ is placed at distance $\frac{c}{10}$ from sensor $s_i$; similarly, the five last sensors $s_{12l_{i,j}-8+c}^{i,j}$ for $1 \le c \le 5$ are placed at distances $\frac{c}{10}$ from sensor $s_j$. As for the remaining sensors, sensor $s_{5+c}^{i,j}$ is placed at distance $\frac{1}{2} + \frac{c}{12}$ (distance defined as Euclidean distance traveled when going along the line segments of the edge) from sensor $s_i$ for $0 \le c \le 12l_{i,j} - 8$. With this distribution, each line segment end point of the edge is also a sensor position. Let $s_{p_1}^{i,j}, \ldots, s_{p_{|p|}}^{i,j}$ be sensor positions on line segment end points of edge $(v_i, v_j)$ (except for $s_i$ and $s_j$). We slightly refine the positions of sensors $s_{p_t-2}^{i,j}$ and $s_{p_t+2}^{i,j}$ for $1 \le t \le |p|$ in order to make the reduction work: $s_{p_t-2}^{i,j}$ is placed at distance $\frac{1}{2} + \frac{p_t-2}{12} - \frac{1}{60}$ from sensor $s_i$ and sensor $s_{p_t+2}^{i,j}$ is placed at distance $\frac{1}{2} + \frac{p_t+2}{12} + \frac{1}{60}$ from sensor $s_i$ for $1 \le t \le |p|$. To complete the reduction, we set the unit radius of all sensors to $\frac{1}{10}$, parameter $k'$ (the number of relays to be placed) of instance $I'$ to $k + \sum_{(i,j)\in E}(4l_{i,j} - 1)$ and the energy threshold $E_T = n'$.

We need to show that $I'$ is a YES-instance if and only if $I$ is a YES-instance. We first show the "if"-direction: if a dominating set of cardinality $k$ exists in graph $G$, there exists a solution for $I'$ with $k'$ relays and a total energy use of exactly $E_T$. Let the dominating set in $I$ be $S = \{v_1, \ldots, v_k\}$ w.l.o.g. Then the solution $S'$ for $I'$ places $k$ relays at sensor positions $s_1, \ldots, s_k$; moreover, it places $4l_{i,j} - 1$ additional relays on sensor positions for each edge $(i, j)$ with $i < j$ (to avoid double-counting on the bidirectional edges) as follows: if $s_i \in S'$, then relays are positioned at sensor positions $s_{3c}^{i,j}$ for $1 \le c \le l_{i,j}$; if neither $s_i$ nor $s_j$ are in $S'$, then the relays are positioned at sensor positions $s_{2+3c}^{i,j}$ for $0 \le c \le 4l_{i,j} - 2$. Thus, we have positioned $k'$ relays in such a way that each of the $n'$ sensors can reach a relay in a single hop, resulting in an overall en-

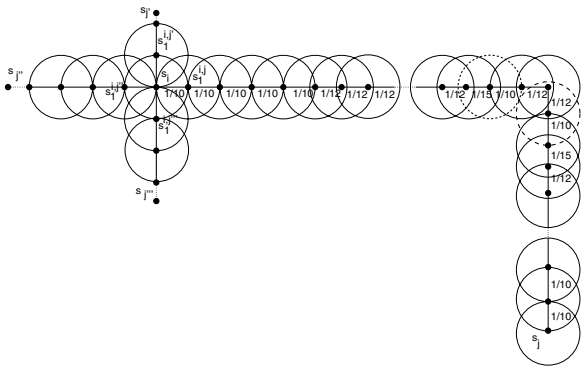


**Figure 2. Illustration of the reduction: for the "only if"-direction each cell needs to be checked. Note that the two dashed circles do not intersect.**

ergy use of $n'$, as required.

In order to show the "only if"-direction, we assume that a solution $S'$ with total energy use at most $E_T$ is given as a set of relay positions. We modify the solution in such a way that lets us read off a solution $S$ for $I$ as follows: each relay in $S'$ that is not placed at a sensor position is moved to a sensor position from which it still hears all the sensors (plus possible additional ones) that it heard before the move. The construction and the parameter choice make this feasible. To see this, consider Figure 2, which shows a small part of the construction. A case analysis for each cell of Figure 2 shows that relays can always be moved without changing the set of sensors that they hear and therefore also without changing the total energy $E_S$ of the whole system, which is at most $E_T = n'$. Obviously each of the $n'$ sensors must use at least one hop to transmit its packet to a relay (even if they are co-located), therefore no solution for $I'$ can exist with $E_S < n'$. Also, if a sensor requires two or more hops to reach a relay in any solution, the solution must use more energy than $n'$. Thus, our modified solution must be a dominating set for the unit disk graph of $I'$. Due to the way that sensor positions are distributed on the edges of the original graph $G$, an edge $(i, j)$ of length $l_{i,j}$ will be covered by $4l_{i,j} - 1$ relays sitting on internal sensor positions. The construction is such that $4l_{i,j} - 1$ relays suffice to cover all internal sensor positions $s_1^{i,j}, \ldots, s_{12l_{i,j}-3}^{i,j}$ but not $s_i$ or $s_j$ unless one of the two has a relay as well. If $s_i$ have a relay on them, then $4l_{i,j} - 1$ relays always be placed such that $s_j$ is covered as well (and vice-versa). Thus, we can read off a solution for $I$ by just including in the dominating set $S$ all nodes $v_i$ that have a relay on the corresponding node $s_i$ in $S'$. ■

## 3. Heuristic Approaches

We consider four heuristic approaches to solve the RELAY POSITIONING problem. Since the RELAY POSITIONING problem can, in general, be viewed as a clustering problem, the main idea for each approach is taken from known clustering algorithms. The problem can be rendered as grouping the sensors into clusters, and assigning each cluster one forwarding relay in such a way as to minimize the cost of such a solution. The cost function used to measure a quality of a solution is the sum of shortest path-lengths (measured in hop-count) of sensors to their corresponding relays (in order to reduce parameter space for our experiments, we assume that all sensors need to deliver about the same amount of data to the base station, which corresponds to setting all weights $f_i = 1$ in Eq. 1); thus, cost defined this way corresponds to the total energy use of the network.

The basic approaches are described here, followed by details about each algorithm. In the following, $k$ is the number of relays we want to position, and $n$ is the number of sensors. We remind the reader that we are only discussing *algorithms* on how to compute the final relay positions; thus, when we say that an algorithm "places a relay and then updates its position", the relay will have to be physically moved only once, namely when its *final* position has been computed (see Step 6 of the meta-algorithm).

**a) k-means:** The $k$ relays are initially placed in randomly selected POSITIONS and $k$ clusters are created by assigning sensors to the nearest (in hop-count) relay. The algorithm then iterates as follows until no changes are made during an iteration:

- For each relay, find a position which minimizes its DISTANCE to the sensors in its cluster.
- Recompute all clusters (for each sensor, find the relay that is closest in hop count). This may result in some nodes changing clusters they belong to.

**b) k-global:** The $k$ relays are again initially placed in randomly selected POSITIONS, and an iteration consists of the folowing step that is repeated until the solution no longer changes:

- For each relay (in random order), find the POSITION that minimizes the cost of the overall solution given that all other relay positions remain unchanged (i.e. the sum of lengths of the shortest paths from all sensors to their nearest relay is minimized).

**c) bottom-up:** This algorithm starts by placing relays in the same positions as the nodes, thus creating $n$ relays and $n$ one-sensor clusters. It then runs the following steps $n - k$ times:

- Find two clusters with the smallest DISTANCE between the relays.
- Combine the two clusters by deleting one relay and placing the other one in a position that minimizes DISTANCES between the relay and sensors in the combined cluster.
- Recompute all clusters

**d) greedy:** This approach places relays one-by-one in $k$ steps. In each step, it finds a POSITION for the new relay such that the cost of the whole solution is minimized, given that positions of the relays already placed remain unchanged.

We use two different ways to define POSITIONS and DISTANCE, creating sixteen implementations of the above approaches. POSITIONS is a finite set generated from the locations of the sensors, and "finding a POSITION" means searching through this set. We used two ways to define POSITIONS: all positions of the sensors (with a size of $n$) and all positions of the sensors plus all intersection points of the sensors' radii (size of $O(n^2)$). The reason for trying both sets is that while the first is relatively small and widely used by hierarchical sensor network algorithms [3], using the intersection points may result in better solutions. In fact, there always exists an optimal solution with relays placed in the second POSITIONS set, as shown in Section 2.

DISTANCE is a metric used to compute distances between relays and sensors in certain parts of the algorithms. We again use two ways to define it: as Euclidean distance and as a hop-count distance. In the k-means algorithm, using the Euclidean distance metric results in finding the average position of the sensors in the cluster, while the hop-count metric requires a search through a given POSITIONS set. The rationale behind trying both metrics is again an apparent trade-off: while Euclidean metric is very simple to compute, it may not at all reflect the actual hop-count distance of two sensors.

We have chosen eight of the 16 algorithms are (the naming scheme is <approach>-<metric>-<whether radii intersection points were considered>) for experiments; we list them including their theoretical running times, where $n$ is the number of sensors, $k$ is the number of relays, and $D$ is the average degree of the sensors (i.e., a sensor has $D$ other sensors as neighbors on average):

- `kmeans-euk` with a running time of $O(kn)$ per iteration
- `kmeans-hops-no` with a running time of $O(n^2D + knD)$ per iteration
- `kmeans-hops-yes` with a running time of $O(n^3D + knD)$ per iteration
- `kglobal- hops-no` with a running time of $O(kn^2D)$ per iteration
- `kglobal-hops-yes` with a running time of $O(kn^3D)$ per iteration
- `bottomup-euk` with a running time of $O((n^2 + nD)(n - k))$
- `greedy-hops-no` with a running time of $O(kn^2D)$
- `greedy-hops-yes` with a running time of $O(kn^3D)$

Note that `greedy-hops-yes` is guaranteed to find a solution that is at most a factor $\frac{e}{e-1} = 1.58$ off the optimum solution, thus we can only expect relatively small improvements over this algorithm. The running times are obtained by straight-forward analysis; for `kmeans` and `kglobal`, we can give an upper bound of $O(n^2)$ on the number of iterations, since the worst possible solution will have at most $n$ paths (one from each sensor) of length $O(n)$ and we only proceed to the next iteration if the algorithm finds a better solution. These upper bounds will clearly never be reached in reality, but they prove that our algorithms have polynomial running times.

## 4. Simulation Results

The scenarios we use to compare the eight algorithms in simulations all have 1024 sensors spread on a square field. The transmission ranges are the same for all sensors and are set to values that result in expected average degrees of 6, 12, 18 and 24 in the connectivity network. The number of relays to be used is 1, 2, 4, 8, 16, 32, 64 and 128. When the algorithms start, the sensors are placed uniformly at random on the field first. Then a relay is placed in the middle of the field and only sensors that are able to connect to the relay are considered for the actual run of the algorithm (if less than 90% of the sensors are connected, the sensors are positioned again). This corresponds to Steps 1 through 4 of the meta-algorithm in section 1. Consequently, the k-means and k-global algorithms only place the remaining $k - 1$ relays randomly.

For each scenario, we ran the algorithms and collected the cost of the resulting solutions and running times of the algorithms. We ran each scenario multiple times with different random seeds until the standard deviation of the cost was about 10% of its mean.

The results are shown in the following figures. Figure 3 shows the resulting cost of a solution for different number of relays and average network degree of 6. The figure shows that the costs found by the algorithms are remarkably similar (the worst is within 15% of the best). The difference decreases with increasing number of relays, and almost disappears when the number of relays gets to 128, where near-optimal solution is found by all algorithms (majority of sensors are connected directly to relays). The best algorithm is, nonetheless, the `kglobal-hops-yes`.

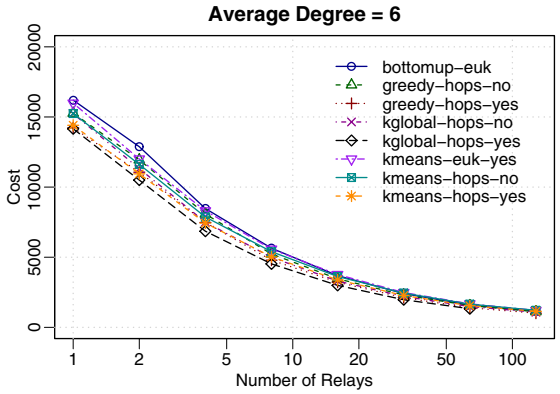


**Figure 3. Cost of a solution for various number of relays (on log scale) and all algorithms for a topology with an average degree of 6.**

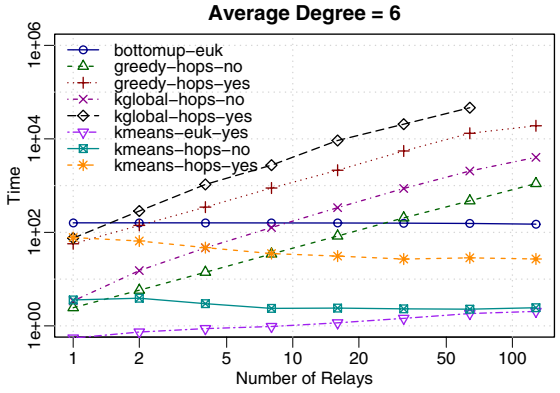


**Figure 4. Running time (in [s] on log scale) for various number of relays (on log scale) and all algorithms for a topology with an average degree of 6.**

Figure 4 again shows the scenario with average degree of 6 and all algorithms, but now the y-axis is the time taken for one run to be completed (notice the log scale of y-axis). This figure shows two very different behavior classes with respect to running time of the algorithms. One scales as a power-law: for `greedy` and `kglobal`, this finding is consistent with the theoretical running times for these algorithms, which depend linearly on $k$. Similarly, `kmeans-euk-yes` exhibits power-law behavior, which again is consistent with its theoretical runing time; relative to `greedy` and `kglobal`, the slope seems to be smaller, which can be explained through the different number of iterations that this algorithm must make to converge.

The running time of the second class of algorithms

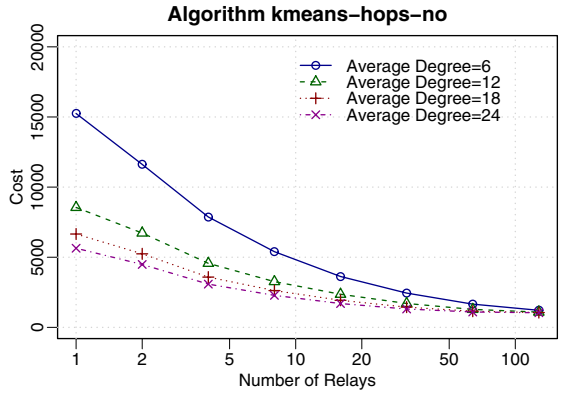


**Figure 5. Cost of a solution for various number of relays (on log scale) and different average degrees for the `kmeans-hops-no` algorithm.**

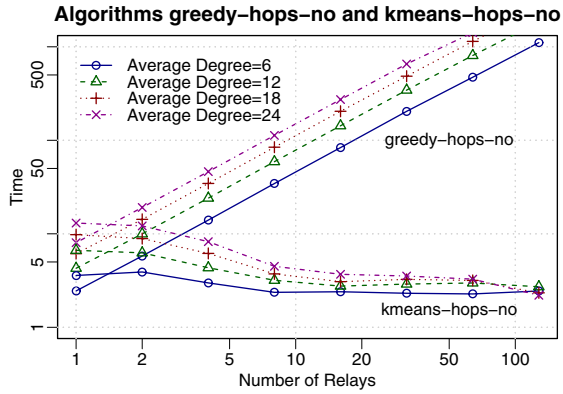


**Figure 6. Running time (in [s] on log scale) for various number of relays (on log scale) and different average degrees for the `kmeans-hops-no` and `greedy-hops-no` algorithms.**

is insensitive to or even declines with the number of relays. In the case of `kmeans-hops-yes` and `kmeans-hops-no`, it seems that the low-order term containing $k$ in the theoretical running time is outweighed by the fewer iterations that the algorithm has to go through with a larger number of relays, thus resulting in a small time decrease for larger number of relays. Thus, `kmeans-hops` seems to converge faster when we have relatively small clusters. Finally, `bottomup-euk`'s running time also very slightly decreases with an increasing number of relays, which again is consistent with its theoretical running time. The `kglobal-hops-yes` algorithm is

clearly the worst in running time. The `kmeans-hops-no`, though still very competitive in the cost of solutions found, has a much smaller running time.

Figure 5 shows how the cost of solutions varies with average node degree of the connectivity network. The higher the connectivity, the lower the cost, as expected. The curves meet for 128 relays again, confirming that for 8 sensors per relay on average, a near-optimal solution is found. The other algorithms behave very similarly. Again, these findings are consistent with the theoretical running times of all algorithms, where average degree $D$ is a linear factor. Finally, Figure 6 shows how the running time evolves with connectivity for selected representations of both classes (`greedy-hops-no` and `kmeans-hops-no`).

The key learnings from the simulations can be summarized as follows: using more time-intense algorithms yields results that are a few percentage points better than very fast algorithms. We believe that in most applications the solution quality improvement gained by using more computational power is probably offset by the cost of such additional computational power. In particular, allowing relays to be placed on intersection points allows algorithms to find slightly better solutions at considerable additional computation time.

## 5. Related Work and Conclusion

Hierarchical approaches and related clustering approaches for wireless networks have been studied intensely in the past, see [4] for a survey. See [1] for an introduction to basic concepts in sensor networks. In contrast to previous work, our approach calls for two different types of transceivers, one of which can be moved to specific locations. Our scheme allows for truly simple sensors that never need to serve as cluster-heads with high emission radii and computing power, such as in the scheme of [3], in which it is not known a priori which sensor may have to act as cluster-heads and all sensors must therefore be strong enough to potentially act as cluster-heads.

We believe that our two-phased approach has potential for future research. We are working on a system where the relays find their positions in a self-organizing manner by moving to high traffic volume locations and changing their position as the network load changes. Other future work includes a variation of our approach in which load is distributed fairly and maximum energy use per node is minimized instead of overall network energy use.

## References


[1] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, Erdal Cayirci; *A Survey on Sensor Networks*; IEEE Communications Magazine, August 2002.

[2] A.D. Amis, R. Prakash, T.H.P. Vuong, D.T. Hyunh; *Max-Min D-Cluster Formation in Wireless Ad Hoc Networks*; Proceedings INFOCOM 2000

[3] Suman Banerjee and Samir Khuller; *A Clustering Scheme for Hierarchical Routing in Wireless Networks*; INFOCOM 2001

[4] Y. P. Chen, A. L. Leistman, and J. Liu; *Clustering Algorithms for Ad Hoc Wireless Networks*; in Ad Hoc and Sensor Networks, Edited by Y. Xiao and Y. Pan, Nova Science Publisher, 2004

[5] Cornuejols, G., Fisher, M., and Nemhauser, G.; *Location of bank accounts to optimize float: An analytic study of exact and approximate algorithms*; Management Sci. 23, 789-810, 1977

[6] Garey, M. R. and Johnson, D. S.; *Computers and Intractability: A Guide to the Theory of* NP*-Completeness*; Freeman, San Francisco, 1979

[7] L. Valiant; *Universality Considerations in VLSI Circuits*; IEEE Transactions on Computers, 1981