Intelligent Street Lighting Clustering

Richard Verhoeven
Eindhoven University of Technology
P.O.Box 513
Eindhoven, The Netherlands
p.h.f.m.verhoeven@tue.nl

Natasa Jovanovic
Eindhoven University of Technology
P.O.Box 513
Eindhoven, The Netherlands
n.jovanovic@tue.nl

Johan J. Lukkien
Eindhoven University of Technology
P.O.Box 513
Eindhoven, The Netherlands
j.j.lukkien@tue.nl

ABSTRACT

The advances in dynamic street lighting introduce new functionality for control and maintenance of the street lighting infrastructure. Vital elements in this infrastructure are the powerful controlling devices that control separate groups of light poles and collect information from the system. For an infrastructure based on wireless communication, this paper describes a fast heuristic algorithm for selecting the locations of these controllers and computing their light poles assignments. In addition, we present the analysis of the simulation results obtained by testing our algorithm for six street lighting networks with real geographic locations of their light poles.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Network topology; C.2.1 [**Network Architecture and Design**]: Wireless communication; F.2.2 [**Nonnumerical Algorithms and Problems**]: Routing and layout

Keywords

Citywide; clustering; segmentation; street lighting; wireless network

1. INTRODUCTION

The installation of LED lighting in the street lighting system of a city [1] provides considerable energy saving [2] compared to existing technologies. Since LED technology supports faster and fine-grained control, the introduction of LED luminaires enables more advanced dimming regimes, such as dynamic dimming based on detected activity on the street. The street lighting infrastructure of a city can consist of hundreds of thousands of light poles, which in order to implement dynamic lighting control and run intelligent lighting applications are coupled by outdoor luminaire controllers (OLCs) that enhance the system by providing communication capabilities. Given the size of the street lighting network in large cities, the fully centralized approach of managing such a network is neither desirable nor feasible. A semi-distributed network as the chosen architecture introduces bridging devices called *cluster controllers* (CCs) between individual light poles and the central management system of the lighting operator. In this way, the entire network is divided into a number of non-overlapping clusters of OLCs, where each cluster is managed by one cluster controller.

The size of a cluster of OLCs that can be managed by a single cluster controller depends on the chosen communication technology. Due to the application requirements such as uploading energy consumption statistics, individual pole control and responding to detected activity, powerline communication is often not sufficient for the data exchange among OLCs and between OLCs and cluster controllers [1]. Especially cluster boundaries introduce problems in such a network. By using wireless communication, the disadvantages of powerline communication with respect to cluster boundaries are overcome.

Due to mesh networking and multi-hop communication, the number of OLCs managed by a single cluster controller can be increased from 140 to 200 for powerline communication [3, 4] up to 500 to 4000 [5, 6] for wireless communication. Due to the large number of OLCs per cluster controller, the number of cluster controllers per city is reduced greatly. With a small number of cluster controllers, the positioning of these controllers within a city becomes a complex optimization problem, which should take into account wireless communication properties like radio range, end-to-end delay, bandwidth and reliability and hardware robustness. Moreover, the costs of hardware and installation of the cluster controllers represent usually a significant portion of the total hardware and installation costs, imposing an additional incentive to ensure the number of cluster controllers deployed is as small as possible.

In general, for a given network of light poles (OLCs), the *street lighting clustering* problem addresses three interconnected questions; see Figure 1 for an illustration.

- **Q1:** In how many clusters should we divide the network?
- **Q2:** How to assign light poles, i.e., wireless nodes to clusters?
- **Q3:** Where to place the cluster controllers?

These questions implicitly lead towards the assumption that we can define a multi-objective cost function that we

WiMobCity'14, August 11, 2014, Philadelphia, PA, USA.

ACM 978-1-4503-3036-7/14/08 ...$15.00.
http://dx.doi.org/10.1145/2633661.2633668.

Figure 1: An illustration of a manual clustering of one part of the street lighting network in Eindhoven.

can use to assess a certain clustering regardless of the method that provided it. However, defining such a function to objectively quantify a clustering of a network taking into account all criteria mentioned above represents a challenge on its own. In this paper, we focus on minimizing the average end-to-end delay in the clusters of the network, where the number of clusters is predefined. Hence, we use the average hop distance between the cluster controller and its OLCs as an objective measure of the quality of the clustering. In our current and further work we examine other metrics.

The rest of the paper is organized as follows. We give the formal problem statement in Section 1.1 and present the state-of-the-art clustering methods in Section 1.2. In Section 2 we present the heuristic algorithm that we developed for the street lighting clustering, as well as the time-complexity analysis of that algorithm. The results of the extensive testing of our algorithm in clustering of real street lighting networks with tens of thousands of nodes are presented in Section 3. We conclude the discussion in Section 4.

1.1 System model and problem statement

Let P be a set of n outdoor luminaire controllers given by their geographic locations and let $G = (P, E)$ be a graph representing the network of OLCs, with the set of nodes P and the set of edges E, where an edge in E indicates that its end nodes are within communication range of each other. The street lighting clustering problem can be formally defined as follows.

Given graph G, determine the set $C = \{c_1, c_2, \ldots, c_k\}$ of k cluster controller positions and the assignment $\mathcal{A}$ of each OLC to exactly one of the cluster controllers, such that the average hop distance between a cluster controller and an OLC assigned to that controller is minimized.

In general, the positions of the cluster controllers can be arbitrarily selected inside the area covered by the network. However, for reasons of simplicity, we restrict the positions of the cluster controllers to the positions of OLCs.

The clustering problem we defined above is certainly intractable and determining the set of optimal solutions is not feasible. However, in large scale networks like the street lighting systems, finding the optimal clustering is not of essential importance, especially because it is very difficult to measure what is truly optimal in practice. Therefore, our goal is to find a method that provides a solution that is sufficiently good in terms of the network performance. This automatic method is certainly an improvement over the clustering manually done by humans that is based on objectives such as historic, cultural or administrative divisions, which are irrelevant for the network performance and reliability.

Although an optimal assignment of OLCs to cluster controllers might be computed using a global algorithm, the actual assignment of OLCs to cluster controllers should preferably be done based on local information. In other words, instead of using explicit configuration assignments, each OLC selects a cluster controller based on information about the direct neighborhood. In that case, the system can adjust itself to cope with changing network conditions, such as OLC failures, failing network connections or failing cluster controllers.

1.2 Related work

The optimal placement of cluster controllers is related to the k-means [7] and k-medoids [8] clustering algorithms. The k-means algorithm selects k centers within an n-dimensional space, such that the sum of the squared distances between each data point and its closest selected center is minimal. Often, the Euclidean distance is used as the distance function, although it is possible to use arbitrary distance functions. The k-medoids algorithms are similar to k-means, except that k-medoids selects centers that are actual data points. Both algorithms start with an initial selection of centers and update that selection until a fixed point is reached where no further improvement is possible. The final selection is a local optimum, with limited guarantees in comparison with the global optimum. In particular, the initial selection can have a significant impact on the performance of the final solution [9].

To apply the standard k-medoids algorithm in the street lighting clustering, the Euclidean distance function must be replaced by a hop distance function, which is either computationally expensive or requires considerable preprocessing and an excessive look-up table. The storage size and computation effort to create a complete table of pair-wise distances based on the shortest path in the given graph quickly becomes unmanageable for the expected datasets. Using Floyd-Warshall's algorithm and the transitive closure algorithm, the storage size would be $n \times n$, while the computation time would be in the order of $n^3 \times MaxDistance$. An alternative would be using the Breadth-First Search algorithm, which results in a computation time in the order of $n \times |E|$.

When the positions of CCs are restricted to a limited set, for example due to city regulations, the optimal placement of cluster controllers becomes an instance of the Facility Location problem [10, 11], which is proven to be NP hard. When the number of feasible positions is small, existing algorithms for the FL problem can be applied. However, a cluster controller can be placed in a small cabinet and attached to any light pole, which typically results in a large number of feasible positions, thus affecting the execution time of those algorithms.

When the hop distance between an OLC and a CC is restricted to a maximum, the optimal placement of clus-

ter controllers becomes an instance of the Dominating Set problem. Since this problem is also known to be NP hard, a polynomial exact algorithm is not possible, although approximation algorithms are available [12].

The selection of CC locations and the assignment of OLCs is also related to the clustering of wireless sensor networks. While energy consumption and network longevity represent no issue in the street lighting networks, there are common objectives with WSN regarding the clustering process and the properties of the clusters formed and their cluster heads, i.e., CCs in our case. The taxonomy of different WSN clustering attributes, as well as a survey of a large number of WSN clustering algorithms is presented in [13]. Other related work in this domain includes [14, 15, 16, 17, 18].

Although the street lighting clustering can be regard-ed as instances of other problems for which there are many approximation algorithms, the hop distance function used results in significantly longer execution time compared to the execution time of the same algorithms when the Euclidean distance is used. The heuristic algorithm we present in the next section is designed to deal with the specific constraints implied by the wireless technology used and moreover, it is fast enough that on top of it, we can build an interactive interface that can provide solutions for a given manual input or that can compute multiple metrics that would allow comparison between different clusterings. In addition, since the algorithm does not require an extensive preprocessing step or a large look-up table, it is possible to manipulate the graph to evaluate different network conditions.

2. STREET LIGHTING CLUSTERING ALGORITHM

The street lighting clustering algorithm is an iterative algorithm, where in each iteration

1. locations of k cluster controllers are selected, and
2. nodes(OLCs) are assigned to cluster controllers, i.e., grouped into clusters, based on the currently selected locations of cluster controllers.

In more detail, for a selection of cluster controller locations $\{c_1, \ldots, c_k\}$, each node is assigned to the cluster controller that is on the smallest hop distance from that node. The hop distances between a specific node and the cluster controllers can be computed for instance, using Dijkstra's shortest path algorithm. Note that this node assignment procedure based on the shortest hop distance results in the connectedness of each of the clusters separately. In addition, after the assignment of nodes to the cluster controllers is completed, the average hop distance in each cluster is easily computed since the hop distances are already determined during the assignment phase.

The set of CC locations in one iteration is selected such that it represents an improvement over the set of locations from the previous iteration. More precisely, in the j-th iteration of the clustering algorithm, the assignment $\mathcal{A}_i^{(j)}$ of nodes to the cluster controller on location $s_i^{(j)}$ results in average hop distance $HopDist_i^{(j)}$ between the nodes of that cluster and their cluster controller. Within the set of node locations of the assignment $\mathcal{A}_i^{(j)}$, we search for a CC location that would result in average hop distance of that cluster being smaller than $HopDist_i^{(j)}$. If such a location is found,

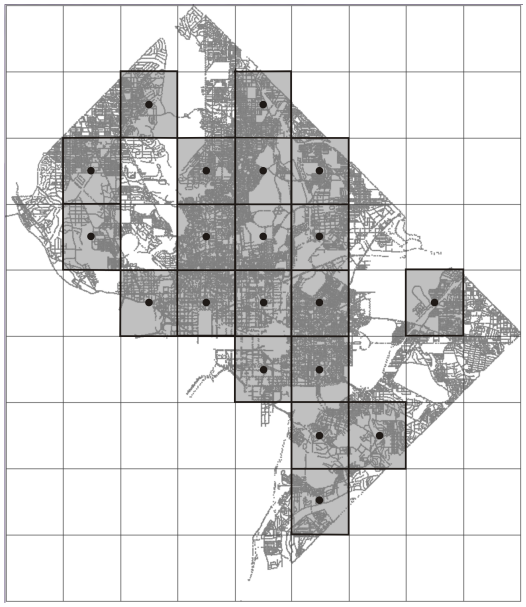

Figure 2: The initial selection of cluster controller locations.

then that location becomes a selected CC location $c_i^{(j+1)}$ in iteration $j+1$, otherwise, the old location is kept, i.e., $c_i^{(j+1)} = c_i^{(j)}$. This search for a better cluster controller location regarded as an optimization step, is done for all the clusters in each iteration. The algorithm ends when for none of the clusters a better CC location can be found.

It is important to note that unlike in the k-medoids algorithms, we do not search for the optimal location of the cluster controller within the set of nodes assigned to it. A complete search would require the pair-wise shortest paths for the nodes within that cluster, which is still computationally intensive. Instead, we search for a better cluster controller location within a limited set of node locations that includes the centroid of the cluster and the local neighborhood of the current CC location. The search is started by computing the average hop distance within the cluster if the cluster controller is placed on the node location that is the closest to the geographic centroid of the cluster. If this location does not provide an improvement, then the search is continued within the local neighborhood of the current CC location. This local neighborhood in the beginning consists of about 25 neighboring nodes and it only gets expanded if there was no change in CC location in two consecutive iterations of the main loop. Finally, this means that in the last iteration of the algorithm, the "local" neighborhood would expand to the point that it contains all the neighbors of the cluster controller from the previous iteration.

The initial selection of locations for the placement of cluster controllers can be done in many different ways. While a random selection of k locations may be the easiest way, such a selection should be avoided since there are no restrictions in the choice of locations. When randomly selected, the locations may be very close to each other, which would consequently result in either very slow convergence or a very uneven clustering. There is a possibility to choose the initial locations manually, however, for a fully automatic clustering of the network, we use algorithm 1 that selects locations

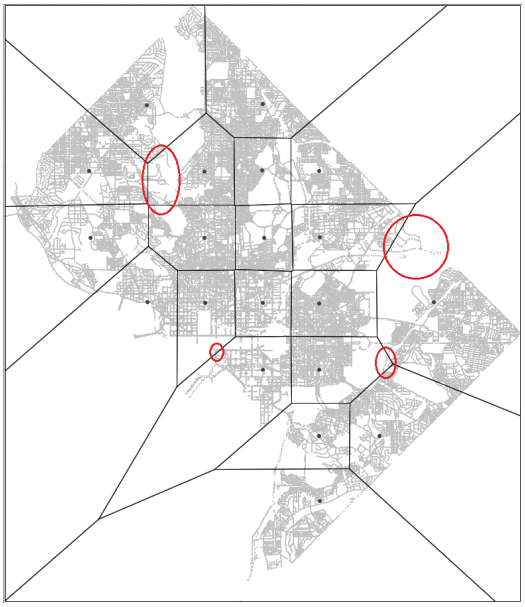

Figure 3: The clustering of the Washington DC street lighting network based on Euclidean distance results in four out of twenty clusters being disconnected.

based on the density of nodes in the network that are evenly distributed on the geographic map.

Algorithm 1 INITIAL CONTROLLERS PLACEMENT

```
function InitialControllersPlacement(P)
    Determine the smallest bounding box B of P;
    Split B into m × m rectangular grid;
    Determine the k densest unit boxes B_1, ..., B_k;
    for i ← 1 to k do
        Find node p_c closest to the center of B_i
        c_i ← p_c
    end for
    return c_1, ..., c_k
end function
```

For a set of node locations given by their coordinates in a geographic map, it is easy to determine the smallest bounding box of that map, i.e., the smallest rectangle that contains all node locations. The bounding box is then divided into a rectangular grid by splitting it into $m \times m$ identical unit boxes (rectangles); see Figure 2. The number of unit boxes must be at least equal to the number of clusters, thus, the number m of rows and columns in the rectangular grid is chosen such that $k \leq m^2$. For each unit box in the grid, we determine the number of node locations that it contains. From the total of m^2 unit boxes, we choose the k densest ones for the initial placement of cluster controllers. More precisely, the selected k locations are the k node locations closest to the centers of the densest boxes. Different choice of the number m of rows and columns in the grid results naturally in different sets of initial locations for the cluster controllers. For small values of m, the cluster controllers are guaranteed to be positioned far from each other. With increasing of m, the density of nodes becomes the dominating factor allowing the cluster controllers to be positioned close to each other.

Below we present the pseudo-code of the clustering algorithm. As previously stated, the input of the clustering algorithm is the connectivity graph G representing the network of n nodes. For a chosen number k of clusters, the output of the algorithm is the set C of cluster controller locations and the corresponding assignment $\mathcal{A}$ of the nodes to the cluster controllers. Since the hop distances between a set of CC locations and all other nodes in G can be computed in $\mathcal{O}(n + |E|)$, the worst-case time complexity of the clustering algorithm is $\mathcal{O}(k(n + |E|))$.

The final remark in this section we make in order to justify the use of computationally demanding hop distance metric instead of a fairly simple Euclidean distance. Namely, after the selection of initial cluster controller locations, the clustering based on Euclidean distance would result in a two-dimensional Voronoi diagram [19] with k Voronoi cells, which can be computed in $\mathcal{O}(k \log k)$ time. Determining the subsets of nodes that corresponds to each Voronoi cell can be done in $\mathcal{O}(kn)$ time, which is practically linear in the number of nodes, since $k \ll n$. However, this clustering can result in disconnected clusters (see Figure 3), since the small geographic distance between a cluster controller and a node does not imply the connectivity between the two. Therefore, the clustering based on Euclidean distance is not practically useful in the domain of multi-hop wireless networks.

Algorithm 2 CLUSTERING ALGORITHM

```
while Improvement detected do
    if C is empty then
        C ← InitialControllersPlacement(P)
    else
        for j ← 1 to k do
            Compute the hop sum σ between
            node p_c closest to the centroid of A_j
            and all other nodes in A_j
            if σ < HopSum(A_j) then
                c_j ← p_c                ▷ Improvement detected
            else
                Compute local neighborhood N of c_j;
                Find a node p̄ ∈ N such that
                hop sum σ between that node and
                the nodes in A_j is minimized;
                if σ < HopSum(A_j) then
                    c_j ← p̄              ▷ Improvement detected
                end if
            end if
        end for
        if No improvements and size(N) < max then
            Increase size(N)             ▷ Improvement detected
        end if
    end if
    Initialize assignments A_1, ..., A_k, where c_j ∈ A_j
    and HopSum(A_j) = 0, j = 1..k;
    Using a breadth-first search algorithm
    starting from c_1, ..., c_k,
    compute for each p_i, i = 1..n the nearest c_j, j = 1..k
    and HopDist between p_i and its nearest c_j;
    A_j ← p_i
    Increase HopSum(A_j) for HopDist
end while
```

3. SIMULATION RESULTS

Using the clustering algorithm we presented in the previous section, we derived clusterings of six street lighting networks of the cities of Amersfoort, Eindhoven and Rotterdam in the Netherlands and the cities of Los Angeles, San Diego and Washington DC in the USA. For these cities, maps with the GPS coordinates of their light poles are publicly available, hence, we used them to test our clustering algorithm and explore the attributes of the computed clusters.

In a real deployment of the street lighting system, objects like buildings, vehicles and trees influence the communication range and the actual hop distances between the nodes can be accurately determined only after the system of OLCs is deployed. Since the lack of deployments does not allow us to collect the real data, we calculated the hop distances between the nodes using a simple radio communication model with a specific communication range. More precisely, the connectivity graph G for each of the six cities is determined assuming the one hop distance to be any Euclidean distance that is smaller than the given communication range r, i.e., $E = \{e(i,j) |\ i,j \in P, d(i,j) < r\}$, where d is the Euclidean distance between a pair of nodes.

From the available datasets on the street lighting locations, we have extracted the largest connected subsets to test the clustering algorithm, assuming a communication range of 300 m (based on measurements using the 868MHz frequency). The clustering algorithm assumes the input graph (map) to be connected. There are well-known methods to identify separate connected subgraphs of an arbitrary graph, and this can be done in a preprocessing step. After the connected subsets are identified, the clustering algorithm can be used for subsets of size larger than 4000 nodes, since any smaller subset can be assigned to a separate cluster controller.

The size of the testing datasets ranges from 19448 nodes in the largest connected subset of the Los Angeles network to 94713 nodes in the largest connected subset of the Rotterdam network. Assuming that a cluster controller can support a maximum of 4000 nodes, the lower bound k_{min} on the number of clusters can be determined for each of the datasets, i.e., $k_{min} = \lceil \frac{n}{4000} \rceil$. We used the clustering algorithm to derive clusterings with different numbers of clusters (cluster controllers), starting from k_{min} and increasing that number to at most $2 \cdot k_{min}$. Since increasing the number of clusters results in decreasing the average cluster size, an increase of k beyond $2 \cdot k_{min}$ would result in the waste of hardware capabilities and significant increase in all hardware related costs. In addition, minimizing the number of cluster controllers while keeping the network operational is one of the main incentives behind the street lighting clustering.

Table 1: The minimum average hop distance in the clustering of Los Angeles (19448 nodes)

Clusters	Avg hop dist	Max cluster size
5	8.23	4584
6	7.36	4646
7	6.71	3869
8	6.42	3869
9	6.10	3461
10	5.70	3137

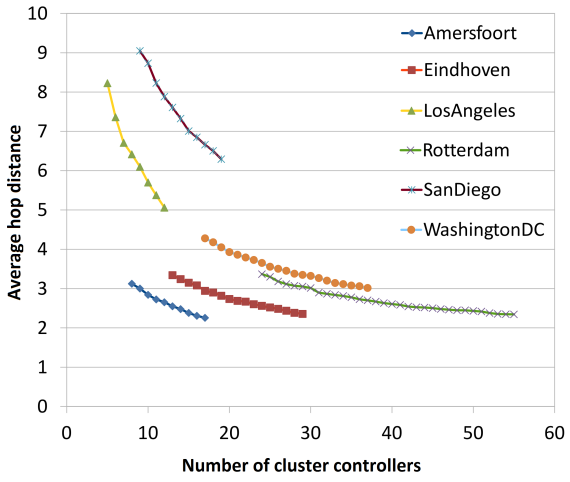

Figure 4: The average hop distances in clusterings of the street lighting networks of Amersfoort, Eindhoven, Los Angeles, Rotterdam, San Diego and Washington DC.

Table 2: The minimum average hop distance in the clustering of Amersfoort (28666 nodes)

Clusters	Avg hop dist	Max cluster size
8	3.12	5009
9	3.00	5018
10	2.84	3924
11	2.72	3764
12	2.65	3748
13	2.55	2919
14	2.48	3297
15	2.38	3297
16	2.31	2485

Figure 4 and Table 1-6 present the average hop distances of different clusterings derived using our clustering algorithm on the six different datasets. From the results, it is easy to notice that the average hop distance is a monotonically decreasing function of the number of clusters. Higher hop distances in the clusterings of Los Angeles and San Diego in comparison to the other four cities are the consequence of larger distances between the light poles. For the cities of Amersfoort, Eindhoven, Rotterdam and Washington DC, the algorithm provides clusterings with relatively low average hop distances, usually around 3 hops with a small number of cluster controllers.

Table 1-6 also indicates the size of the largest cluster in the corresponding clustering with the minimum average hop distance. Although the average cluster size decreases with increasing the number of clusters, the maximum cluster size can still be larger in clusterings with more clusters than in the ones with less clusters. Furthermore, the results indicate that in many cases some of the clusters are of size larger than 4000 nodes. This is the consequence of an uneven distribution of light poles in a city, with the downtown center typically containing more light poles than residential areas, parks and industrial zones. Since the clustering algorithm assigns a light pole to the nearest CC (measured in hops)

Figure 5: Two different clusterings of the street lighting network of Los Angeles, as a result of two different initial sets of 7 cluster controller positions.

Table 7: Total execution time and execution times of separate phases of the clustering algorithm given in seconds and computed as an average over different number of clusters.

City	Poles	Edges	Pole assign.	First iteration	Iteration, $\|\mathcal{N}\|=25$	Iteration, $\|\mathcal{N}\|=100$	Iteration, $\|\mathcal{N}\|=400$	Avg. #iter.	Total execution
Los Angeles	19448	1149374	0.009	0.020	0.155	0.407	0.376	14.80	3.68
Amersfoort	28666	8168068	0.028	0.438	0.919	3.343	9.641	16.33	51.27
San Diego	33136	2246090	0.017	0.045	0.297	0.793	2.265	16.50	10.69
Eindhoven	50516	12964066	0.040	0.735	1.300	4.655	12.791	16.86	67.49
Washington DC	66094	10835358	0.064	0.455	1.568	5.627	10.198	22.96	105.76
Rotterdam	94713	28659169	0.131	1.561	3.909	13.784	44.479	21.79	345.83

Table 3: The minimum average hop distance in the clustering of San Diego (33136 nodes)

Clusters	Avg hop dist	Max cluster size
9	9.05	7242
10	8.73	5450
11	8.23	5277
12	7.88	5390
13	7.61	5823
14	7.32	5012
15	7.01	5025
16	6.85	5025
17	6.66	5030
18	6.50	5004

Table 4: The minimum average hop distance in the clustering of Eindhoven (50516 nodes)

Clusters	Avg hop dist	Max cluster size
13	3.34	6019
14	3.24	6126
15	3.15	5742
16	3.08	4379
17	2.94	4191
18	2.90	4454
19	2.82	4441
20	2.73	3320
21	2.69	3717
22	2.67	3837

regardless of the cluster size, large clusters are formed in the dense city areas. It is a matter of future work to investigate how the restrictions on cluster size would affect the average hop distance. Another related question that rises is the one of comparing visually quite different clusterings with a marginal difference in average hop distance, like the two clusterings in Figure 5. Future research should identify all metrics relevant to the network clustering, in which case we could give objective assessments of different clusterings considering multiple metrics.

Finally, in Table 7 we present the average execution times of different phases of the clustering algorithm measured in seconds, obtained by running the clustering algorithm on a PC with an $i5$ processor on 3.2 GHz and a cache of 4 MB. The average is computed over all computed clusterings of one city for different number of clusters. As indicated earlier, the execution times are proportional to the number of edges in the graph. For example, this can be seen in comparison of the execution times for the cities of Amersfoort and San Diego. Although Amersfoort has nearly 4500 nodes less than San Diego, the number of edges is more than 3.5 times larger, resulting in considerably larger execution times of separate iterations and consequently, 5 times larger total execution time, where the average number of iterations for both cities is the same. More importantly, the results indicate short running times of the pole assignment phase and the first iteration of the clustering algorithm, the former being mostly under 100 msec and the latter mostly under 1 sec, with slightly larger values obtained only for the Rotterdam dataset. As expected, the optimization phase takes the largest part in the total execution times, where the running of an iteration including the local search within 400 neighbors can take up to 12 times more time than the running of an iteration with the local search within 25 neighbors.

Table 5: The minimum average hop distance in the clustering of Washington DC (66094 nodes)

Clusters	Avg hop dist	Max cluster size
17	4.28	5396
18	4.18	5401
19	4.05	5397
20	3.93	5409
21	3.86	5331
22	3.79	5295
23	3.73	5161
24	3.65	4530
25	3.56	5327
26	3.50	5165

Table 6: The minimum average hop distance in the clustering of Rotterdam (94713 nodes)

Clusters	Avg hop dist	Max cluster size
24	3.37	7034
25	3.30	6264
26	3.18	6264
27	3.11	6264
28	3.07	6415
29	3.05	6415
30	3.01	5837
31	2.90	4736
32	2.87	5173
33	2.84	5173

Our tests also showed that $70-90\%$ of the time is spent on achieving an improvement in average hop distance of less then 2%, and in rare situations this improvement can be at most 10%. In other words, increasing the size of the local neighborhood beyond 25 neighbors does not result in a significant improvement in the average hop distance. Using small neighborhoods for the local search reduces the total execution time to just a few seconds. Short running time allows the clustering algorithm to be incorporated in an interactive tool that can be used to compute any desired number of clusterings and provide insights into the network performance that is essential in the planning and deployment of the intelligent street lighting system.

To verify the efficiency of the presented algorithm, the performance is compared with an implementation of the k-medoids algorithm in C, for which several metrics are presented in Table 8. For the construction of the pair-wise distance table, the breadth-first search algorithm is used which has a complexity $n \times |E|$. For the selected datasets, the execution time for table construction varies from a few minutes to several hours, which is not suitable for the earlier mentioned interactive tools. The presented table size is based on distance values that can be represented with single bytes. For hop distances, that representation is sufficient for the selected datasets. In the implemented k-medoids algorithm, every iteration will search and select the best cluster controller from each cluster. The computation time for each iteration depends on the number of clusters and its complexity is in the order of $k \times n + n^2/k$ (under the assumption that clusters have similar sizes). Since k is selected based on the average number p of poles per cluster, the complexity can be rewritten to $n^2/p + n \times p$. The execution times for the iterations are presented for the two extreme values for k. Since the execution times are independent of the number of edges, the performance is more predictable. The average number of iterations is determined by selecting random sets of k initial cluster controllers and applying the k-medoids algorithm to determine a stable local optimum solutions. The number of iterations in this implementation is smaller than the number of iterations in the algorithm we suggested in Section 2, which can be addressed to the faster convergence of the full search. The average execution time is based on the same data as the average number of iterations. Although the execution time for the optimization part of the k-medoids algorithm is better, the execution time for the construction of the pair-wise distance table should be included in overall execution time of the algorithm. With information about the mode of operation, it is unclear how often the computed distance table can be reused. When the graph changes regularly, the reuse of the table will be limited.

For an interactive tool where an engineer or architect is exploring the possible solutions for different network configurations, the original k-medoids algorithm is less suitable due to excessive preprocessing step to compute the pair-wise distance table. In addition, the distance table results in serious hardware requirements, especially when the method should be applied to even larger cities. The suggested algorithm has no excessive memory requirements other than the input graph and no preprocessing steps other than loading the graph or computing it. In an interactive tool, the algorithm is able to present results immediately after it is started.

4. CONCLUSION

To allow dynamic control and monitoring of the intelligent street lighting, the network of wirelessly connected luminaire controllers must be divided into clusters managed by cluster controllers for which the suitable locations should be chosen. As a solution to this challenging optimization problem, we presented a fast heuristic algorithm that uses average hop distance in the clusters as an objective metric of the network performance to compute a clustering with a given number of clusters. The short execution times measured when running the algorithm on six street lighting networks of different size and topology indicate that the algorithm is applicable in practice.

5. REFERENCES

[1] C. Atici, T. Ozcelebi and J. J. Lukkien, *Exploring user-centered intelligent road lighting design: a road map and future research directions.* Consumer Electronics, IEEE Transactions on, vol.57, no.2, pp. 788–793, 2011.

[2] E-streetlight Project *Guide for energy efficient street lighting installations*, 2007. http://www.e-streetlight.com/Documents/Homepage

[3] Philips *LFC7065 Segment Controller*, 2010. http://www.lighting.philips.com/pwc_li/main/products/controls/assets/lfc7065ds.pdf

[4] Echelon *CPD 3000 Outdoor Lighting Controller* 2013. https://www.echelon.com/products/components/docs/CPD_3000.pdf

[5] Tvilight *Tvilight Intelligent Street Lighting*, 2013.

Table 8: Execution time for the k-medoids algorithm given in seconds and computed as an average over different number of clusters.

City	Poles	Table constr.	Table size	Iteration $k = \lfloor n/2000 \rfloor$	Iteration $k = \lfloor n/4000 \rfloor$	Avg. #iter.	Avg. execution
Los Angeles	19448	0:02:25	360 MB	0.052	0.108	7.86	0.58
Amersfoort	28666	0:15:45	784 MB	0.086	0.143	7.33	0.81
San Diego	33136	0:07:27	1.02 GB	0.098	0.169	11.93	1.49
Eindhoven	50516	0:41:09	2.48 GB	0.154	0.256	8.11	1.59
Washington DC	66094	1:06:41	4.07 GB	0.253	0.415	10.96	3.35
Rotterdam	94713	3:51:22	8.35 GB	0.341	0.557	10.53	4.45

http://www.tvilight.com/wp-content/uploads/2013/09/tvilight-brochure-en.pdf

[6] Philips *LFC7300 Starsense Wireless Segment Controller Kit Datasheet*, 2012. http://www.lighting.philips.com/pwc_li/main/products/controls/assets/sc-kit-lfc7300-datasheet.pdf

[7] J. B. MacQueen, *Some methods for classification and analysis of multivariate observations*. Proc. of 5th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297, 1967.

[8] L. Kaufman and P. J. Rousseeuw, *Clustering by means of Medoids*. Statistical Data Analysis Based on the L_1–Norm and Related Methods, ed. Y. Dodge, North-Holland, 405–416, 1987.

[9] J. M. Peña, J. A. Lozano and P. Larrañaga, *An empirical comparison of four initialization methods for the K-Means algorithm*. Pattern Recognition Letters, vol.20, Issue 10, pp. 1027–1040, 1999.

[10] H. Pirkul and V. Jayaraman *A multi-commodity, multi-plant, capacitated facility location problem: formulation and efficient heuristic solution*. Computers & Operations Research, vol.25, Issue 10, pp. 869–878, 1998.

[11] S. Li, *A 1.488 approximation algorithm for the uncapacitated facility location problem*. Information and Computation, vol.222, pp. 45–58, 2013.

[12] F. V. Fomin, D. Kratsch and G. J. Woeginger, *Exact (exponential) algorithms for the dominating set problem*. Graph-Theoretic Concepts in Computer Science, Lecture Notes in Computer Science vol.3353, pp. 245–256, 2005.

[13] A. A. Abbasi and M. Younis, *A survey on clustering algorithms for wireless sensor networks*. In Computer Communications 30(14), pp. 2826–2841, 2007.

[14] S. Banerjee and S. Khuller, *A clustering scheme for hierarchical control in multi-hop wireless networks*. In Proc. of IEEE INFOCOM 2001, Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 2, pp. 1028–1037, 2001.

[15] E. Ilker Oyman and C. Ersoy, *Multiple sink network design problem in large scale wireless sensor networks*. In Proc. of IEEE Intl. conf. on Communications, vol. 6, pp. 3663–3667, 2004.

[16] K. Akkaya, F. Senel and B. McLaughlan, *Clustering of wireless sensor and actor networks based on sensor distribution and connectivity*. Journal of Parallel and Distributed Computing, 69(6), pp. 573–587, 2009.

[17] F. G. Nocetti, J. S. Gonzalez and I. Stojmenovic, *Connectivity based k-hop clustering in wireless networks*. Telecommunication systems, 22(1-4), pp. 205–220, 2003.

[18] S. Yang, J. Wu and J. Cao, *Connected k-hop clustering in ad hoc networks*. In Proc. of IEEE Intl. conf. on Parallel Processing, pp. 373–380, 2005.

[19] M. de Berg, M. van Kreveld, M. Overmars and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*. Springer, 2000.