Clustering problem in self-organising networks on the example of smart vehicle transportation system

Marek Antosiewicz[a], Przemysław Szufel[b], Bogumił Kamiński[c], Agata Skorupka[d], Paweł Prałat[e], Atefeh Mashatan[f]

Abstract. The emergence of the Internet of Things and Intelligent Transportation System (ITS) opened doors for innovative solutions in the transportation area, such as smart vehicles and Vehicular Networks (VN). These solutions provide an opportunity to increase efficiency, safety and reduce traffic and the negative impact the transportation sector has on the environment. These trends are associated with challenges around the rapidly increasing number of vehicles. In this paper, we formulate an optimisation graph clustering problem with an objective function consistent with optimising the efficiency of the network understood as its throughput or the packet delivery/loss ratio (but can be generalised to other cases). Due to the constantly increasing size of VN, the optimisation algorithm needs to be

[a] SGH Warsaw School of Economics, Collegium of Economic Analysis, Al. Niepodległości 162, 02-554 Warszawa, Poland, e-mail: mantosi@sgh.waw.pl, ORCID: https://orcid.org/0000-0003-3035-8591.
[b] SGH Warsaw School of Economics, Collegium of Economic Analysis, Al. Niepodległości 162, 02-554 Warszawa, Poland, e-mail: pszufe@sgh.waw.pl, ORCID: https://orcid.org/0000-0001-9525-3497.
[c] SGH Warsaw School of Economics, Collegium of Economic Analysis, Al. Niepodległości 162, 02-554 Warszawa, Poland, e-mail: bkamins@sgh.waw.pl, ORCID: https://orcid.org/0000-0002-0678-282X.
[d] SGH Warsaw School of Economics, Collegium of Economic Analysis, Al. Niepodległości 162, 02-554 Warszawa, Poland, e-mail: agata.skorupka@gmail.com, ORCID: https://orcid.org/0000-0003-4487-8758.
[e] Toronto Metropolitan University, Department of Mathematics, 350 Victoria St., Toronto, ON, Canada, M5B 2K3 e-mail: pralat@torontomu.ca, ORCID: https://orcid.org/0000-0001-8898-8778
[f] Toronto Metropolitan University, Department of Information Technology Management, 55 Dundas Street West, Toronto, ON M5G 2C3 e-mail: amashatan@torontomu.ca, ORCID: https://orcid.org/0000-0001-9448-9123

computationally efficient. Next, we adapt classic community detection algorithms (CDAs), i.e. the Louvain and ECG CDAs, to our objective function, and assess their performance. We find that graph CDAs provide slightly worse results in terms of objective function value, but they are significantly faster than other studied approaches. Clearly, there is a trade-off between these two, but taking into account the computation efficiency, especially given large networks and the fact that the objective function deterioration is slight, graph CDA performance seems satisfactory, and they may be useful in solving similar problems.
Keywords: communication networks, network design, graph clustering
JEL: DL85, C61, L90

1. Introduction

The efficient exchange of information and communication are critical factors for the operational efficiency of various network systems. Across many types of networks, natural subgroups (clusters) of identities continually form and dissolve. The communication within such clusters is relatively efficient, while in-between cluster communication incurs larger costs. One example could be a social network representing the communication within a department vs. cross-department communication (Creswick et al., 2009). Another example could be the telecommunication network. For example, in TCP/IP networks, subnets can be formed and the observed latency within a single subnet is lower than in cross-subnet communication (O'Malley and Peterson, 1992). In the case of social or technical networks with such properties, it is often possible to decide how to partition the network into subgroups (e.g., organising departments in a company or the logical design of a TCP/IP network). Hence, the optimisation of the communication within such networks requires clustering of network nodes (agents) into logical subnets.

In this paper, we focus on the clustering problem in a smart vehicle network. The widespread use of IoT technologies as well as the emergence of Vehicular

Networks of connected smart vehicles requires a new pattern of communication between them. This standard should take into consideration the optimisation of the information workflow, which can be defined according to the problem. For example, one can mention reducing latency or maximising the quality of service (QoS). For this reason, agents or objects that serve as nodes in the network can gather in subnets, also understood as groups. One of the main challenges in this field is the constantly growing size of the network, which necessitates algorithms that are computationally efficient. In our work, we use the example of a smart cars transportation system, but our problem formulation can be applied to both social networks and artificial networks such as the Internet of Things (IoT).

The literature on the subject focuses on the following three areas: (1) Intelligent Transportation System, (2) Internet of Vehicles and (3) Vehicular ad-hoc Networks.

As mentioned above, the spread of connected autonomous vehicles (smart cars) requires a new pattern of communication between them, known in the literature as the Intelligent Transportation System (ITS). The ITS is an interconnected system using advanced technologies and providing innovative and reliable applications that facilitate collaboration between its stakeholders. The ITS aims at reducing risks (i.e. accidents, collisions and thefts), traffic congestion and energy use, while at the same time maximising comfort and satisfaction for the end users (Qureshi and Abdullah, 2013).

Current developments in the technology of smart cities and the Internet of Things (IoT) intensify communication between users as well as between users and infrastructure (often acting as an intermediary), which leads to a significant increase in the amount of data transferred. The above-mentioned challenge requires not only reliable network systems supporting the appropriate patterns of communication, but also the applications of big data techniques and big data architectural solutions (including cloud computing). This challenge has led to the development of the of Internet of Vehicles (IoV) concept (Kaiwartya et al., 2016, Coppola and Morisio, 2016, Lu et al., 2014).

One of the core ideas of the IoV is Vehicular ad-hoc Networks (VANET), i.e. the formulation of a network supporting different types of communication (Soares and Rodrigues, 2021, Hbaieb et al., 2021), namely V2V (vehicle to vehicle), V2I (vehicle to infrastructure) and V2R (vehicle to road).

VANET is a system based on a network of vehicles equipped with sensors and communicating with each other (using infrastructure, mainly cellular network base stations) in a wireless manner (Badis and Rachedi, 2015, Paul et al., 2017). The VANET concept is an extension of Mobile ad-hoc Networks (MANET), and therefore it 'inherits' MANET's main features, such as dynamic topology due to the high node mobility and the need for scalable solutions as the number of vehicles increases (Rath et al., 2019). On the other hand, it is characterised by unique constraints, i.e. road topology or traffic speed limitations and the possibility of suffering from the packet loss. Nevertheless, VANET is not affected by power or resources constraints. In our problem, we analyse a VANET network where vehicles are communicating with each other using infrastructure, which in this case are cellular network base stations. The nodes in the graph in the following analysis can be substituted with the interpretation of smart cars directly (or other actors in any social or technical network). Therefore, the following problem can be described as 'vehicle to infrastructure' (V2I).

Another feature of the VANET is the ability of nodes to self-organise in subnets (Yick et al., 2008). Therefore, clustering nodes is not only a crucial concept in VANET (Mukhtaruzzaman and Atiquzzaman, 2020), but also a developing area of research in the broader field of networks in general (Shahraki et al., 2020, Kamiński et al., 2021).

Finding the optimal way of dividing network nodes into subnets can serve different objectives:

- minimising the use of power (not applicable to VANET);
- maximising efficient resource utilisation;
- maximising quality of service (QoS, e.g., minimising data transfer loss or latency);
- network load balancing;
- minimising cost or maximising profit.

Clustering schemes can be divided according to the method in the following way:

- intelligence-based strategies;
- mobility-based strategies;
- multi-hop based strategies.

Due to the nature of VNs, the focus in that area is on computationally efficient, as well as scalable, algorithms. Therefore, we focus on the first group, as any algorithm from the second or third group does not compare with intelligence-based strategies in terms of computing efficiency.

Among intelligence-based strategies, one can distinguish machine learning methods and fuzzy logic algorithms. While the latter are characterised by a higher stability of clustering, they are not very computationally effective.

Machine learning algorithms perform better in this regard, but they also have significant shortcomings. First of all, k-means algorithm requires one to know the number of clusters in advance, which makes the solution suboptimal (or requires a high number of runs, which in turn makes it less efficient). Secondly, it is highly sensitive to starting points (initial centroids), so it is not stable. Another commonly used machine learning algorithm is hierarchical clustering, the implementation of which is far more complex (requires a proximity matrix calculation and results in $O(N^2\log(N))$ to $)(N^3)$ time complexity).

Optimisation techniques are used in the problem of network clustering as well; however, they have been mostly concentrated on energy consumption. In these approaches, the wireless sensor nodes are battery-powered, and hence the use of energy is minimised, or the time until the first sensor runs out of energy is maximised. The best-known applications in this area are the following algorithms: LEACH 9 (Heinzelman et al., 2000), HEED (Younis and Fahmy, 2004) and FLOC (Demirbas et al., 2004). These approaches cannot be applied to the problem of VANET, as it is not concerned with the minimisation of energy use.

The goal of the study described in this paper is to apply the clustering problem and community detection algorithms to optimising VN's efficiency. Before, this research problem was mostly analysed in the literature using the game theory approach (Sun et al., 2020), while in other areas, such as other types of networks, community detection and optimisation metaheuristics were successfully used, especially thanks to their short computation time. Furthermore, most literature surveys in the field of network clustering indicate the dependency of clustering methods on the objective function (Shahraki et al., 2020). In this research, we define the clustering problem in a way which is objective function agnostic, that is our approach will work with a wide variety of objective functions. In particular, parameters in an objective function (weights) subject to optimisation during the clustering process may be defined and interpreted depending on the problem and the type of network. In our example, weights represent (or alternatively, are correlated with) the efficiency of the network data transmission (packet loss ratio subject to minimisation). Nevertheless, they can represent any cost of within-group and between-groups communication, therefore may be suitable for a broader class of social network problems.

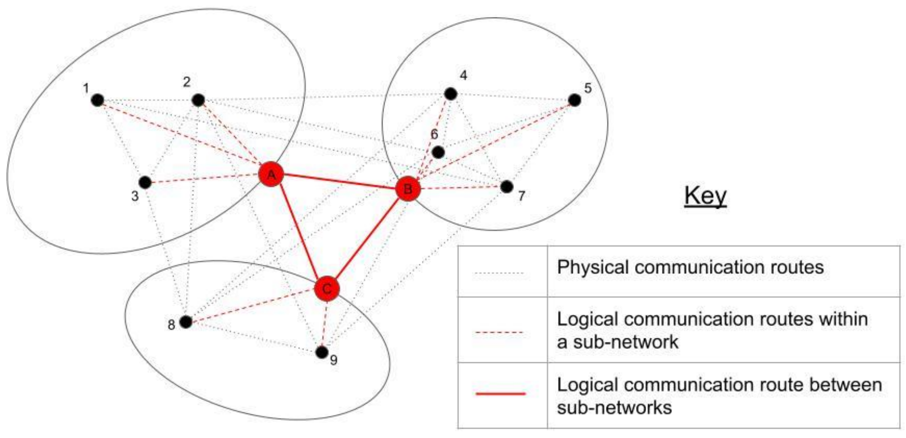
2. Model

In this section, we present the formulation optimisation model for the efficient communication within intelligent transportation networks. In the first subsection, we provide a general non-technical description of the problem, which introduces the next subsection, where we present the formal mathematical optimisation model.

2.1. Description

In our setup, we consider a weighted undirected graph, which consists of nodes and edges that connect them. We assume that the graph is connected, that is, there exists a path between any pair of nodes. The nodes of the graph represent physical cellular base networks through which vehicles are communicating with each other. The edges represent the flow of communication or information between the nodes, with the weights being the amount of communication flow in a unit of time (based on either real-time or historical data, depending on the architecture).

Figure 1. Sample network and division into clusters


1
2
3
4
5
6
7
8
9
A
B
C
Key
Physical communication routes
Logical communication routes within a sub-network
Logical communication route between sub-networks

Source: authors' work.

The graph represents the physical network and it shows how much information needs to be transmitted between the nodes. The network and the physical connections are represented as black dots and black dotted lines in Figure 1. We assume that the actual communication between the nodes takes place through a logical network, which is built on top of the physical network as a logical layer (Esteves et al., 2013).

Creating the logical network involves clustering (grouping) particular nodes of the physical network into distinct subnetworks (Gerla and Kleinrock, 1977). In the example presented in Figure 1, the black nodes represent the physical network, whereas logical subnetworks are marked by black ellipses. In our example, there are three of them. The communication between the nodes in the logical subnetwork takes place through logical nodes, shown as red circles, which are located at the edges of the ellipses. The efficiency of the communication between any two nodes depends on whether they are located within the same subnetwork or not, and the overall efficiency of the network depends on the structure of the logical network, as well as on the clustering algorithm used to perform the grouping (Tai et al., 2011). Given the constant increase of nodes in the network, there is also a need to take computation efficiency into account and choose an algorithm which will be able to solve a task of increasing complexity in a feasible time.

As stated before, the aim of our study is to find an optimal clustering of nodes into logical groups (subnetworks) in order to optimise the efficiency of communication or, in other words, the cost associated with communication. To do this, we first need to propose an objective function which will measure the efficiency of the logical network, and then develop an algorithm which will find the optimal network. The objective function should reflect the following considerations regarding the efficiency of communication for nodes belonging to the same or to different subnetworks. For nodes belonging to the same cluster (subnetwork), the communication is conducted from one of the nodes, to the red logical hub and then back to the other node along the logical connections portrayed by the red dotted lines. The more nodes in the cluster, the more time will be required for the communication to take place. Therefore, we assume that the efficiency of communication for such nodes will be lower when the weight and the size of the cluster is high. For nodes belonging to different logical groups, the communication is conducted from the node to the logical hub along the dotted red line, then to the logical hub of the other node along the solid red line, and finally along the dotted red line to the target node. If the logical network consists of a large number of clusters (subnetworks), the communication time between logical hubs is long. Therefore, the efficiency of communication between nodes belonging to different clusters is a decreasing function of the weight, size, and the total number of clusters.

The mathematical formulation of the proposed objective function that measures the efficiency of the network is described in the next subsection. The measure is consistent with the throughput of the network or with the packet delivery/loss ratio (but may be generalised to any type of cost communication, depending on the type of network). We assume that the efficiency of communication between nodes depends on the following three statistics: a) the volume of communication (weights) within clusters, b) the volume of communication between clusters, and c) the distribution of cluster sizes (in particular, the number of clusters).

2.2. Mathematical formulation

Let us consider an undirected, connected, and weighted graph $G = (V, E)$ having $n = |V|$ nodes representing physical cellular base network stations and $m = |E|$ edges representing desired information flows between stations. Let $w : E \rightarrow R_+$ be the weight function; each edge $e \in E$ has weight $w(e) \in R_+$ associated with it. The Table contains a list of mathematical symbols used throughout the document.

Let P be a partition of the set of nodes V, that is, $P = (V_1, \dots, V_k)$ for some $k \in N$ such that $V_i \subseteq V$ for $i \in [k]$, $V = V_1 \cup \dots \cup V_k$, and $V_i \cap V_j = \emptyset$ if $1 \leq i < j \leq k$. We assume that the efficiency c associated with partition P (for a given graph G and weight function w of its edges) is not affected by any external parameters, that is, $c = c(G, w, P)$. The overall objective in our study is to find the partition P which minimises the cost function $c(G, w, P)$:

$$\min_{\mathrm{P}} c(G, w, P). \tag{1}$$

We restrict our study to some natural family of functions, still very large and flexible, that covers most of the functions one needs to deal with in practice. However, we are looking for a specification of the cost function for which the recalculation of the value will not be computationally intensive given the algorithms that we use for finding the optimum. In particular, the algorithms operate in such a way that the current partition which is being considered is modified in a minor, local way, usually by changing the part for a single node. If we assumed a complex, highly nonlinear cost function, we would need to recalculate the cost function for the entire graph, which for a large graph would be a costly operation. In order to avoid this, we opted for a cost function for which we can update the objective function by performing calculations only for a part of the cost function which corresponds to the parts that are being modified in a single step. Keeping this in mind, we propose the following cost-function specification.

Let $s = (s_1, \dots, s_k)$ be the distribution of part sizes in partition P, that is, $s_i = |V_i|$ for $i = [k]$. Let $\boldsymbol{t} = (t_1, \dots, t_k)$ be the distribution of weights within parts, that is, t_i is the total weight associated with the edges in graph $G[V_i]$ induced by a set of nodes V_i. Finally, let $u = (u_{1,2}, u_{1,3}, \dots, u_{k-1,k})$ be the distribution of weights between parts, that is, u_i $(1 \leq i < j \leq k)$ is the total weight associated with the edges between part i and part j. Note that

$$W = W(w) \coloneqq \sum_{e \in E} w(e) = \sum_{i=1}^{k} t_k + \sum_{1 \leq i < j \leq k} u_{i,j}, \tag{2}$$

that is, each weight is captured by either $\boldsymbol{t}$ or $\boldsymbol{u}$. Moreover, vectors $\boldsymbol{s}$ and $\boldsymbol{t}$ have k coordinates each and vector $\boldsymbol{u}$ has $\binom{k}{2}$ coordinates. Please note that all three vectors together have only $\frac{k(k+3)}{2} = \Theta(k^2)$ coordinates, but they capture a lot of structure of a given partition P. We will restrict ourselves to efficiency functions that only depend on these three vectors, that is, $c = c(\boldsymbol{s}, \boldsymbol{t}, \boldsymbol{u})$, where $\boldsymbol{s} = \boldsymbol{s}(\boldsymbol{G}, \boldsymbol{w}, \boldsymbol{P})$, $\boldsymbol{t} = \boldsymbol{t}(\boldsymbol{G}, \boldsymbol{w}, \boldsymbol{P})$, $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{G}, \boldsymbol{w}, \boldsymbol{P})$.

Table. Symbols used in the document

Symbol	Interpretation
G	graph, $G = (V, E)$ where V and E are the sets of nodes and edges
n	number of nodes, $n = \lvert V \rvert$
v_i	node number $i, v_i \in V$
m	number of edges, $m = \lvert E \rvert$
w	weight function of edges
e_i	edge number $i, e_i \in E$
P	partition of the set of nodes
k	number of parts in partition P
V_i	cluster (part) $i, V_i \in P$
c	efficiency function of partition, $c = c(G, w, P)$
s	distribution of part sizes of partition P

s_i	size of cluster V_i
$\boldsymbol{t}$	distribution of weights within parts
t_i	sum of the weights of edges within cluster V_i
$\boldsymbol{u}$	distribution of weights between parts
$u_{i,j}$	sum of weights of the edges between cluster V_i and V_j
a	weight of within-cluster communication in the cost function
b	weight of the number of parts in the cost function

Source: authors' work.

2.3. Objective functions

Algorithms which will be applied to the optimisation problem further in the paper treat efficiency functions as 'black boxes', i.e. can be applied to any function from that large family and give freedom of interpretation of weights in the graph, according to the specific network. However, there are some natural properties that are typically satisfied in practice. For example, the efficiency usually decreases when the number of parts, k, decreases. Moreover, for a fixed k, the efficiency is typically smaller for unbalanced partitions or, equivalently, vectors $\boldsymbol{s}$.

The functions that we consider in our experiments are similar to the objective function considered by Hallac et el. (2015). We also consider simpler functions that are not highly convex and therefore the value function can be updated quickly, ensuring quick run-time of our algorithms.

2.3.1. Function A: size-insensitive

We assume that the efficiency associated with communication between the parts is fixed, that is, it is proportional to the total weight of edges between the parts. On the other hand, the efficiency of communication between nodes within very small subnets is negligible, but increases with their sizes, and in the extreme case when all the nodes belong to a single network, it is a^2 times larger than the communication between the parts. Therefore, parameter a is responsible for setting the relative cost of within-cluster communication. In this objective function, the cost does not depend explicitly on the distribution of the sizes of parts, i.e. on vector $\boldsymbol{s}$:

$$\mathrm{c_A} = c_A(\boldsymbol{s},\boldsymbol{t},\boldsymbol{u}) = \sum_{i=1}^{k} \left(\frac{at_i}{W}\right)^2 t_i + \sum_{1\le i<j\le k} u_{i,j}, \tag{3}$$

where W is the total weight defined in (2).

2.3.2. Function B: size-sensitive

In this objective function, we take into account the cost associated with the number of subnetworks. In order to keep the specification simple, we do not account for the exact distribution of part sizes but rather the number of parts k, where $k = |s|$. In this formulation, parameter *b* sets the relative cost of creating additional logical subnetworks:

$$\mathrm{c_B} = c_B(\boldsymbol{s},\boldsymbol{t},\boldsymbol{u}) = \sum_{i=1}^{k} \left(\frac{at_i}{W}\right)^2 t_i + \sum_{1\le i<j\le k} u_{i,j} + \left(\frac{k}{b}\right) W, \tag{4}$$

where W is the total weight defined in (2).

3. Algorithms

In this section, we describe the graphs on which we conduct our simulation experiments as well as the algorithm in which we generate them. We concentrate on geometric graphs, since typically in our applications, nodes (base cellular stations) are associated with a geographical location and weights depend on the distance between these locations. Moreover, geometric graphs allow a convenient visualisation. Next, we present the algorithms that we compare which are responsible for finding the optimal clustering of the graphs. We decided to compare two algorithms that were originally proposed for the community detection, namely the Louvain algorithm (Blondel et al., 2008) and the Ensemble Clustering for Graphs (ECG) algorithm (Poulin and Theberge, 2018) with classic metaheuristic approaches, namely the Stochastic Hill Climbing and Simulated Annealing. We also tested a simple greedy and naive algorithm as benchmark solutions.

3.1. Graph generation algorithm

We conduct our simulation experiments on the l–nearest neighbour graph. These graphs have a variety of applications in bioinformatics, data mining, machine learning, manifold learning, clustering analysis and pattern recognition. There are two parameters of the model: n (the number of nodes in a graph) and l (the number of neighbours each node connects to). Our graphs are undirected and weighted, but let us start with a definition of a directed and unweighted auxiliary graph $D = (V, E)$, the l-nearest neighbour graph.

We first generate n nodes forming set V by placing them independently and uniformly at random from the unit square $[0,1]^2$. Then each node $v \in V$ is connected via l directed edges to nodes $u \in V$, where nodes u are the nearest neighbours of v (that is, nodes other than v itself whose distance from v is among the top l smallest ones). Clearly, with probability 1, all pairwise distances are unique, and so the neighbourhoods are well defined, but we can break ties arbitrarily if needed. Note that in this directed graph, each node has out-degree equal to l, but in-degrees have more complex distribution. In particular, it might happen that there is an edge from v to u but not from u to v.

We assign weights to the directed edges of D in the way described below. Each directed edge uv has weight being the Euclidean distance between u and v. Finally, we ignore directions to create an undirected and weighted graph G. Note that if there are two edges, one from u to v and the other one from v to u, then the weights are combined (i.e., the weight of an undirected edge uv is twice the distance between u and v).

It is known that if $l < 0.3043\, ln\, n$, then asymptotically almost surely (a.a.s.) the graph is not connected, whereas if $l > 0.5139\, ln\, n$, then it is connected a.a.s. (Balister and Bollobas, 2005). In our experiments, we take $l = [ln\, n|$ and resample (if needed) to get a connected graph. Let us briefly mention a few algorithmic aspects. The complexity of the trivial brute force method is clearly $O(n^2)$, but it can be easily improved. For example, an $O(n \log^{d-1} n)$ algorithm was presented in (Bentley, 1980), where d was the dimension of the geometric space (in our case, $d = 2$). Clarkson (1983) presented a $O(c^d n\, log\, n)$ algorithm, and Vaidya (1989) showed a worst-case $O((c^r d)^d n\, log\, n)$ algorithm. In our scenario, with $d = 2$ and the nodes generated uniformly at random from the unit square, the graph can be generated even quicker (in linear time) by tessellating the unit square into small squares. By tracking where node v lands, one may investigate the nodes in the same small square and a few neighbouring squares to determine the neighbourhood of v.

3.2. Clustering algorithms

3.2.1. Greedy and naïve algorithms

According to greedy and naive algorithms, the optimal choice is made at each step, without a broader context or the knowledge of previous as well as future steps. Due to their simplicity, these algorithms fail to find globally optimal solution in many classes of problems (e.g. the shortest path). However, in many cases, such a heuristic can approximate a global optimal solution in a feasible time by finding a local optimum.

We designed the greedy algorithm in the following way: in each step we chose to merge two clusters which yield the largest gain in the objective function, whereas the naïve one considers all the possible merging operations of two clusters in a random sequence and merges two clusters if this operation yields an improvement in the objective function.

3.2.2. Louvain

The Louvain algorithm is a variation of a greedy algorithm. Due to the favourable ratio of its quality to speed, it is considered as one of the best algorithms for community detection (Kamiński et al., 2021). The run time of the algorithm is short, $O(n \log^2 n)$, where n is the number of nodes. Compared to other community-detection algorithms, it offers very good quality, but the results are sometimes unstable. When it was introduced in Blondel et al. (2008), the authors showed that it grossly outperformed the existing algorithms both in terms of the quality of results and speed. Furthermore, when comparing it to newer, more advanced algorithms, Louvain still performed remarkably well in both these categories (see Mothe et al., 2017). The operation of the Louvain algorithm can be divided into two steps:

1. The modularity function is optimised locally on all nodes to find communities;
2. The communities are then merged: all the edges within a community are replaced by single weighted loop, resulting in a smaller graph.

Then, the first step is performed on the smaller graph and the algorithm operates until no further increase in the modularity function value is achieved. In our problem, we use a cost function instead of the modularity function.

3.2.3. Ensemble clustering for graphs (ECG)

Ensemble Clustering for Graphs (ECG) is another example of graph community detection algorithms, similar to the Louvain algorithm. As mentioned above, although the latter has strong advantages in terms of results and efficiency, it can produce unstable results. It is attributed to the fact that there are several possibilities resulting in the same change in modularity function value. The Ensemble Clustering for Graphs (ECG) algorithm is aimed at improving these drawbacks by using several runs of the Louvain algorithm and combining them all, in order to produce a more stable outcome.

1. The first phase of the Louvain algorithm is performed l independent times to obtain l partitions of communities;
2. A new version of $G = (V, E)$ is obtained by assigning new weights according to the following formula[1]:

$$w_{uw} = f(x) = \begin{cases} w_* + (1 + w_*)\frac{1}{l}\sum_{i=1}^{l} \alpha_i(u,v), & if\ (u,v) \in C_2(G) \\ & w_*,\ otherwise \end{cases} \quad (5)$$

[1] The *k*-core of graph G, denoted as $Ck(G)$, is the maximal subgraph of G in which all the nodes have at least degree k.

3. The complete Louvain algorithm is applied to G.
Due to executing l independent runs of Louvain algorithms and ensembling them, the ECG algorithm reduces randomness stemming from the fact that the Louvain on its own might produce different solutions in one run, making the algorithm more robust.

3.2.4. Simulated annealing and stochastic hill climbing algorithm

In this subsection, two metaheuristics are presented: stochastic hill climbing algorithm and simulated annealing. Metaheuristics are usually used when a problem is too complex to find a solution using brute force algorithm in the computationally efficient time (Hussain et al., 2019). As they are ‘state of the art’ tools regarding solving similar problems, it would be worth comparing graph clustering algorithms to these metaheuristics. Both metaheuristics make use of randomness during the search process, but while hill climbing is a local algorithm, simulated annealing is a global one.

The stochastic hill climbing is a local search optimisation algorithm that improves the current solution by randomly selecting from available improving moves in the neighborhood of the current solution rather than greedily choosing the best one. Such an approach converges more slowly than steepest ascent methods, but it has a smaller chance of being stuck in a local optimum.

Simulated annealing operates in a similar manner; however, in certain situations it is able to accept a solution which is worse regarding the objective function value (this makes the algorithm a global one). The probability of accepting a worse solution is a decreasing function of the number of steps that have been performed.

4. Numerical efficiency of clustering

4.1. Experiment design

In this section, the simulation setup is described. We test the performance of the algorithms on graph problems of various sizes understood as the number of nodes. We begin with simulating the algorithms on small problems for which we are able to find the global optimum using an exhaustive search method. We calculate the cost function values for $n = \{1, 2, 3, \ldots, 12\}$ using brute force, naive, greedy, Louvain, ECG, uphill and simulated annealing algorithms. For medium-size problems where the number of nodes is equal to $n = \{20, 30, \ldots, 100\}$, we make simulations using the naive, greedy, Louvain, ECG, simulated annealing and uphill algorithms. We assume that $a = 5$ and $b = 20$.

For the large scale problems with $n = 5000$, we test the performance of the Louvain and ECG algorithms using both the objective functions specified in Section 2.3. Furthermore, in order to comprehensively assess the algorithms, we conduct simulations on a grid of parameters governing both objective functions. For the objective function A, we perform simulations for parameter a taking the values from 1 to 10, $a \in \{1, 2, \ldots, 10\}$. For the objective function B, we conduct simulations for all the pairs of parameter values a and b, where $a \in \{1, 2, \ldots, 10\}$ and $b \in \{5, 10, \ldots, 50\}$. In each simulation run we generate the optimal clustering found by the naive, Louvain and ECG algorithms.

We calculate the mean of the objective function over all simulation runs and report the relative performance of the algorithms with respect to the brute force approach (small problems) and the naive approach (medium and large problems). We also assess the stability of the algorithms by analysing the standard deviation of the objective function values found by the algorithms and the computation time of the algorithms.

We run each algorithm 30 times for each graph size and value function parameter and in each run we randomly generate a new graph. During each calculation, the network is constructed on the basis of l–nearest neighbour graph defined in Section 3.1.

4.2. Experiment results

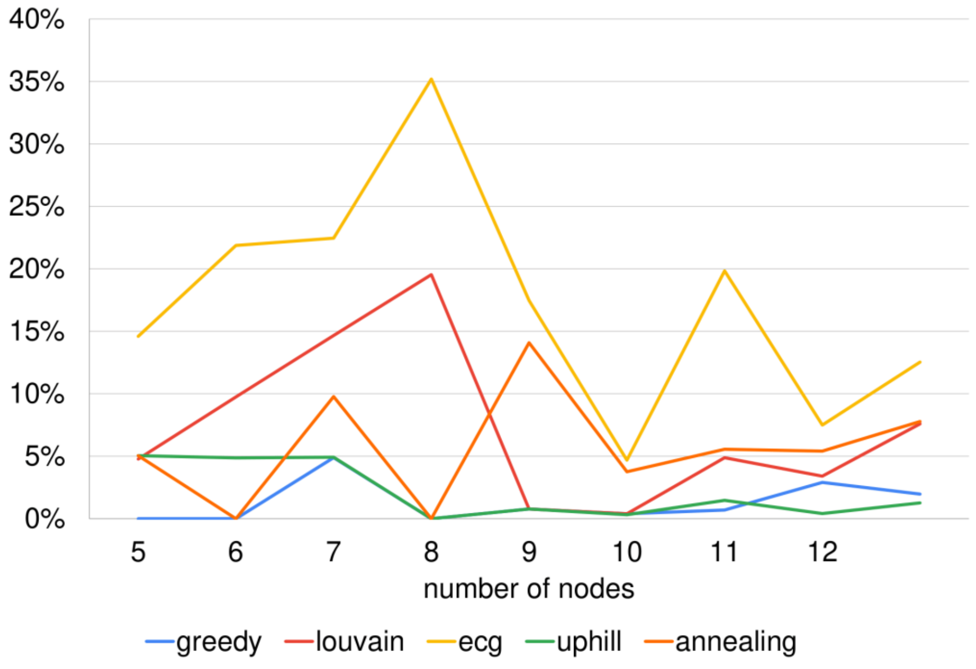
In this section, we present the results of our experiments, i.e. applying clustering algorithms to optimising VNs efficiency problem, including the computing time, as well as objective function values.

4.2.1. Small problems

In this subsection the results of the simulations for small and medium graphs are presented and discussed. We show the deviation of the objective function from the brute force (small problem sizes) and naive approach (medium problem sizes) over the simulation runs for the adopted algorithms, as well as the computation time of the simulations.

In Figure 2, we show the results for the value function in those cases where the number of nodes is small. For these simulation runs, we are able to list all of the possible solutions, and therefore show the global optimal solution. Overall, we concluded that the algorithms which we used yielded good results compared to the global optimal solution provided by the brute force approach. The worst result was obtained by the ECG algorithm for the problem where the number of nodes was equal to 8. The objective function turned out 35% worse than the optimum. That algorithm also tended to provide the worst result for almost all small problem sizes, and additionally it performed worse that the naive approach. The greedy and uphill algorithms tended to yield the best results.

Figure 2. Objective function loss with respect to brute force approach for the algorithms for small number of nodes (in %).

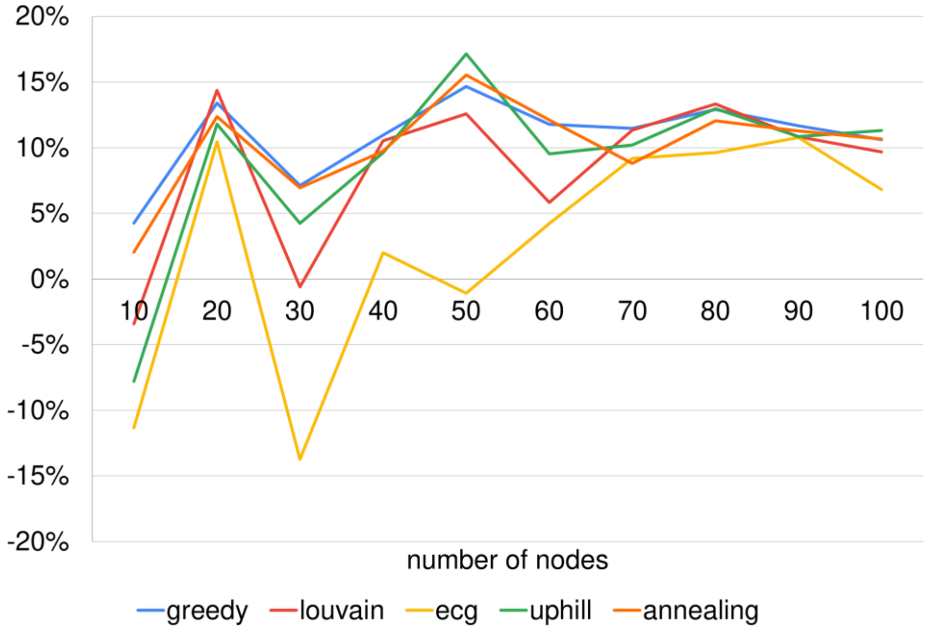


Source: authors' work based on simulation results.

4.2.2. Medium problems

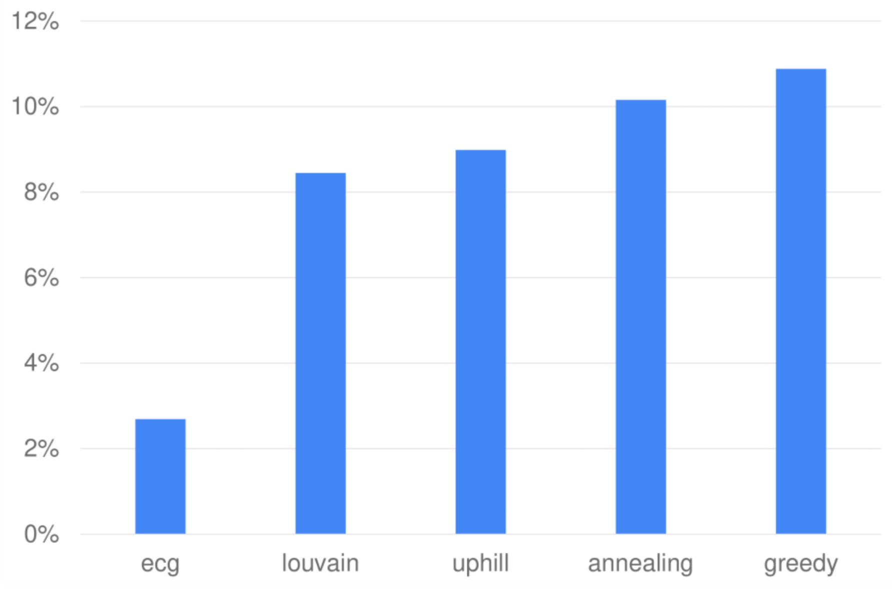
In Figure 3 we show the average resulting value function for problems in which the number of nodes is up to 100. Comparing the results for medium problem sizes gives more accurate information on the performance of the algorithms. None of the algorithms uniformly outperforms all the other ones. Also, we can see that for problem sizes exceeding 50, the naive approach, which is the quickest, yields the worst result. We can observe the same pattern for the ECG algorithm that was visible in its case for small problems: it is the worst-performing formula. In Figure 4, we show the average gain with respect to the naive algorithm over medium problem sizes. This figure confirms the relatively poor performance of the ECG, however the other algorithms also performed relatively badly, with the Louvain showing the average gain of 8.4%. The remaining algorithms show slightly larger gains, e.g. the greedy approach with an average gain of 10.9%. The same results would hold if we limited this analysis to problem sizes exceeding 50 nodes.

Figure 3. Objective function loss with respect to brute force approach for the algorithms for a medium number of nodes (in %)



Source: authors' work based on simulation results.

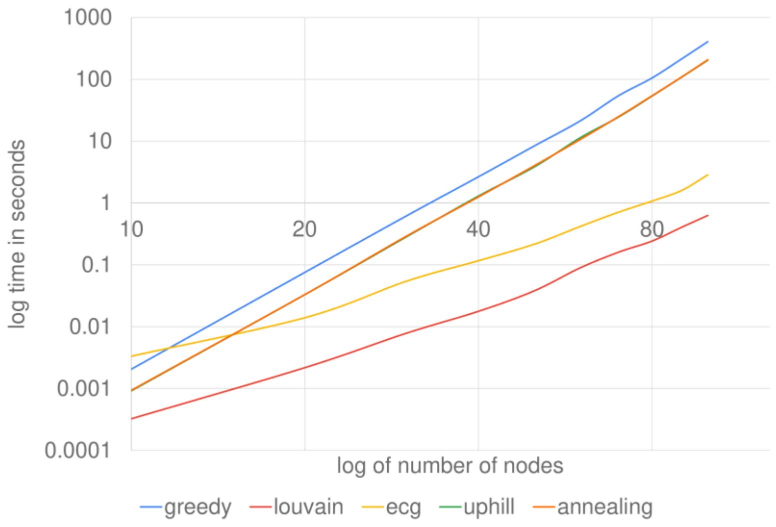
Figure 4. An average percentage gain in objective function value for problems with the number of nodes ranging from 10 to 100 for algorithms.



Source: authors' work based on simulation results.

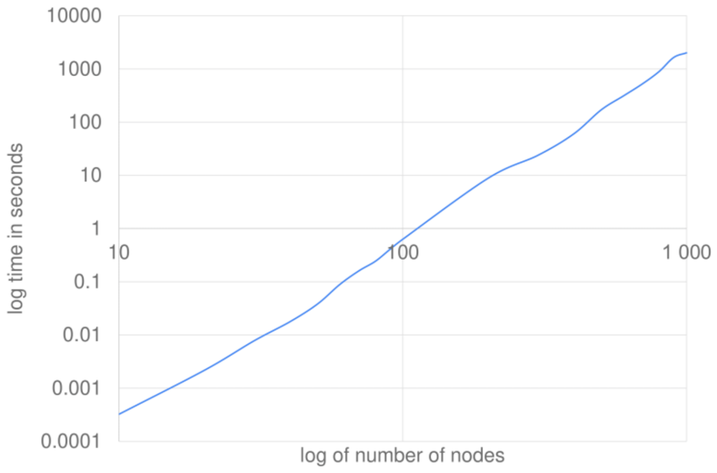
In Figure 5, we show the computation time for the algorithms, and the computation time for the Louvain algorithm for problem sizes of up to 1,000 nodes is presented in Figure 6. The computation times are shown on a log-log plot. We can see that the computation time increases exponentially with the problem size. It is also noticeable that the computation time for both the Louvain algorithm and the ECG are significantly shorter than for the remaining approaches.

Figure 5. Average computation time of algorithms in a log log scale



Source: authors' work based on simulation results.

Figure 6. Average computation time of the Louvain algorithm for a large number of nodes in log log scale



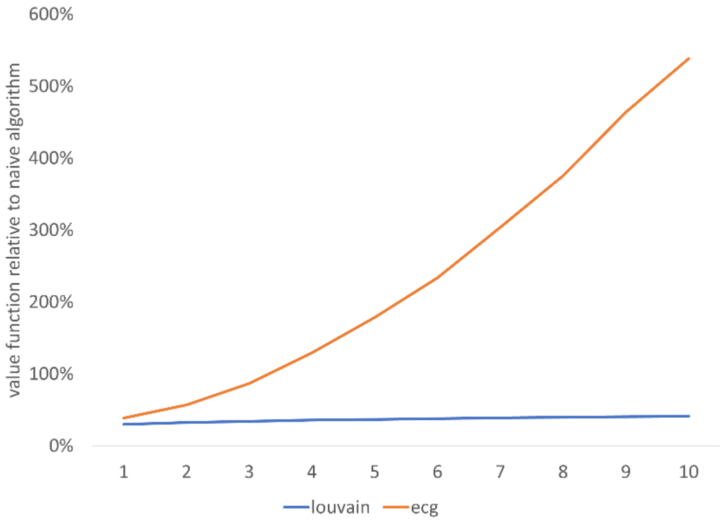
Source: authors' work based on simulation results.

4.2.3. Large problems

We show the results of our simulations in Figures 7–9. In each of the figures, we show the mean objective function value for the algorithm relative to the performance of the naive algorithm. In Figures 8 and 9, darker colours represent lower values, i.e. a better performance of the algorithm in question.

Overall, the community detection algorithms that we used in our study, namely the Louvain and the ECG, show that they perform better than the naive approach, although this is not the case for all parameter values of the objective function. Furthermore, the Louvain algorithm usually performs better than the ECG, unlike in the case of typical community detection problems, where the Louvain algorithm is often unstable.

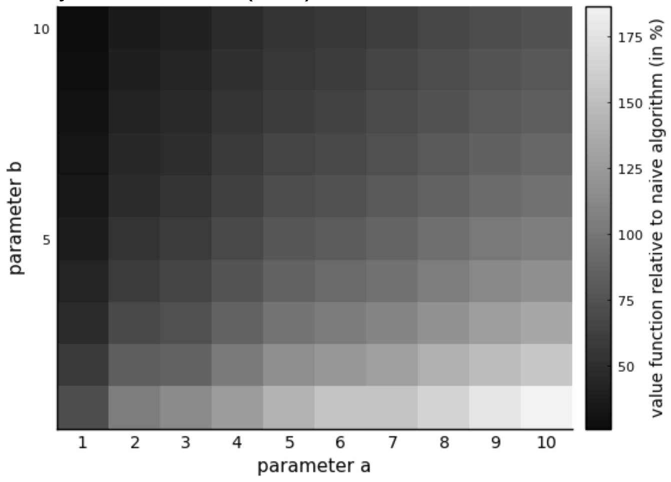
Figure 7. Value function of the Louvain and the ECG algorithms relative to the naive approach for size-insensitive objective function (in %)



Source: authors' work based on simulation results.

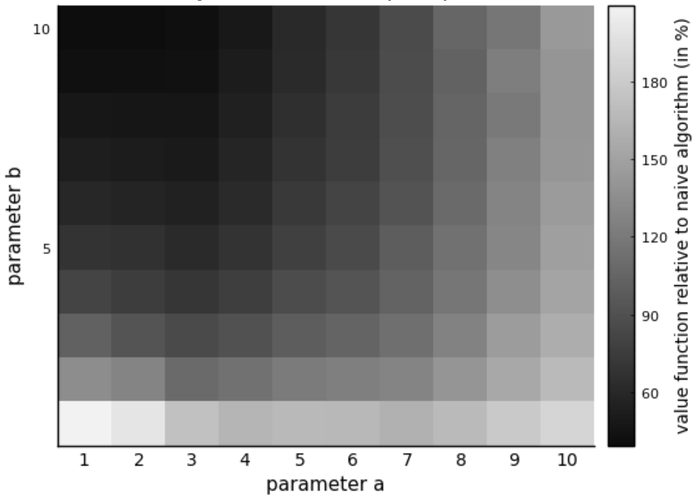
In Figure 7, we show how the two algorithms compare against the naïve approach for the size-insensitive objective function, which is the one where we do not take into account the number of clusters when calculating the objective function. It is visible that the Louvain algorithm provides excellent results across all the parameter values that we simulate. When compared against the naive algorithm, it provides a mean objective value function that is between 59% and 70% better than the naive approach. For smaller values of parameter a, the algorithm provides the best results. Furthermore, the Louvain algorithm completely outperforms the ECG for all the values of a. The ECG algorithm shows satisfactory results only for very small values of a; for values larger than 3 this algorithm performs worse than the naive approach.

Figure 8. Value function of the Louvain algorithm relative to the naive approach for size-sensitive objective function (in %)



Source: authors' work based on simulation results.

Figure 9. Value function of the ECG algorithm relative to the naive approach for size-sensitive objective function (in %)



Source: authors' work based on simulation results.

In Figures 8 and 9, we present how the two algorithms compare against the naive approach for the size-sensitive objective function in which we take into account the number of clusters when calculating the objective function. Since the value function depends on two parameters, we show the results as heatmaps, where darker colours indicate better performance than that of the naive algorithm.

The performance of these two algorithms is more nuanced than is the case for the value function A. First of all, none of them uniformly outperforms the naive approach. The larger the value of parameter a and the smaller the value of parameter b, the poorer their results. Secondly, their performance is of similar quality, with the Louvain algorithm being slightly better than the ECG.

4.2.4. Discussion

Our computational experiments show that classic community detection algorithms, namely the Louvain and the ECG, yield good-quality solutions for the optimal clustering problem in much shorter time than metaheuristic algorithms such as simulated annealing or hill climbing. Furthermore, the latter fail to converge to a good solution in a reasonable time frame for large problems, where the number of nodes equals 5,000. Out of the community detection algorithms, the Louvain algorithm tends to perform better than the ECG, offering solutions with a better value function across most cases that we considered.

It should be noted that some literature on a typical community detection problems indicates the improvement of the ECG algorithm’s performance in relation to that of the Louvain formula and points out that the former alleviates some of the latter’s problems (Poulin and Theberge, 2019). In our setup (which differs from the Louvain in that we do not use modularity function but a different objective function), no improvement in the ECG compared to the Louvain can be observed. On the other hand, in the case of community detection problems, the literature shows that the Louvain algorithm outperforms many classic ones (Mothe et al., 2017), but in our setup it is not always the case. However, it has to be mentioned that it is much faster than other algorithms that we tested. Finally, Hallac et al (2015) consider a similar problem which could be adapted to our setting. They, however, concentrate on highly convex formulations, which could be solved by the Louvain algorithm. They develop the Alternating Direction Method of Multipliers algorithm, which solves the problem efficiently, and which could be applied to a wide array of problems.

5. Conclusions

In this article, we defined a graph clustering optimisation problem. The setup we presented could be applied to a broad range of network routing problems where the network topology changes frequently and where quick and reliable data transmission is necessary. This is the case of vehicular networks, where constantly moving cars change their location in real time, thus changing the topology of the underlying network. An algorithm for clustering such a physical network into a logical one must be computationally efficient, providing good solutions in a short time.

Our study has some limitations, the overcoming of which could become the basis for future research. We considered two specific value functions which represent the quality of the throughput of the network. While the value function specification is general and potentially covers numerous different applications, it is possible that some of them would require a different form of the objective function. However, in such a case, it is possible to simply replace the underlying objective function for the algorithms and run new simulations.

Secondly, we tested the algorithms for a particular network topology generated by the l–nearest neighbour graph. Such a topology mimics the behaviour generated by vehicular networks. This network design does not exhaust all the possible network topologies, and it is not clear how different algorithms would perform for other network types. Testing the performance of community detection algorithms for other networks could be an interesting direction for future research.

Finally, the maximum network size that we consider is 5,000 nodes, while in reality, vehicular networks can consist of tens or even hundreds of thousands of vehicles. Optimising the algorithms to work efficiently for such large networks or developing other procedures to accommodate them could possibly be another problem worth looking into in more detail.

Acknowledgements

This research was funded in part through a generous contribution from NXM Labs Inc as well as the grants from Fields-CQAM and NSERC CRD. NXM's autonomous security technology enables devices, including connected vehicles, to communicate securely with each other and their surroundings without human intervention, while leveraging data at the edge to provide business intelligence and insights. NXM ensures data privacy and integrity by using a novel blockchain-based architecture which enables rapid and regulatory-compliant data monetisation.

References

Badis, H., & Rachedi, A. (2015, April). *Modeling tools to evaluate the performance of wireless multi-hop networks.*

Balister, P., Bollobás, B., Sarkar, A., & Walters, M. (2005). Connectivity of random k-nearest-neighbour graphs. *Advances in Applied Probability, 37*(1), 1–24.

Bentley, J. L. (1980). Multidimensional divide-and-conquer. *Communications of the ACM, 23*(4), 214–229.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment, 2008*(10), P10008.

Clarkson, K. L. (1983). Fast algorithms for the all nearest neighbors problem. In *Proceedings of the 24th Annual Symposium on Foundations of Computer Science (SFCS 1983)* (pp. 226–232). IEEE.

Coppola, R., & Morisio, M. (2016). Connected car: Technologies, issues, future trends. *ACM Computing Surveys (CSUR), 49*(3), 1–36.

Creswick, N., Westbrook, J. I., & Braithwaite, J. (2009). Understanding communication networks in the emergency department. *BMC Health Services Research, 9*, 1–9.

Demirbas, M., Arora, A., Mittal, V., & Kulathumani, V. (2004). Design and analysis of a fast local clustering service for wireless sensor networks. In *First International Conference on Broadband Networks* (pp. 700–709). IEEE.

Esteves, R. P., Granville, L. Z., & Boutaba, R. (2013). On the management of virtual networks. *IEEE Communications Magazine, 51*(7), 80–88.

Gerla, M., & Kleinrock, L. (1977). On the topological design of distributed computer networks. *IEEE Transactions on Communications, 25*(1), 48–60.

Hallac, D., Leskovec, J., & Boyd, S. (2015). Network lasso: Clustering and optimization in large graphs. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 387–396).

Hbaieb, A., Ayed, S., & Fourati, L. (2021, May). *Internet of vehicles and connected smart vehicles communication system towards autonomous driving.*

Heinzelman, W. R., Chandrakasan, A., & Balakrishnan, H. (2000). Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences* (pp. 10–pp). IEEE.

Hussain, K., Mohd Salleh, M. N., Cheng, S., & Shi, Y. (2019). Metaheuristic research: A comprehensive survey. *Artificial Intelligence Review, 52*, 2191–2233.

Kaiwartya, O., Abdullah, A. H., Cao, Y., Altameem, A., Prasad, M., Lin, C.-T., & Liu, X. (2016). Internet of vehicles: Motivation, layered architecture, network model, challenges, and future aspects. *IEEE Access, 4*, 5356–5373.

Kamiński, B., Prałat, P., & Théberge, F. (2021). *Mining complex networks.* CRC Press.

Lancichinetti, A., & Fortunato, S. (2009). Community detection algorithms: A comparative analysis. *Physical Review E, 80*(5), 056117.

Lu, N., Cheng, N., Zhang, N., Shen, X., & Mark, J. W. (2014). Connected vehicles: Solutions and challenges. *IEEE Internet of Things Journal, 1*(4), 289–299.

Mothe, J., Mkhitaryan, K., & Haroutunian, M. (2017). Community detection: Comparison of state of the art algorithms. In *2017 Computer Science and Information Technologies (CSIT)* (pp. 125–129). IEEE.

Mukhtaruzzaman, M., & Atiquzzaman, M. (2020, September). *Clustering in VANET: Algorithms and challenges.*

O'Malley, S. W., & Peterson, L. L. (1992). A dynamic network architecture. *ACM Transactions on Computer Systems (TOCS), 10*(2), 110–143.

Paul, A., Chilamkurti, N., Daniel, A., & Rho, S. (2017, January). Vehicular network as business model in big data (pp. 161–176).

Poulin, V., & Théberge, F. (2018). Ensemble clustering for graphs. In *International Conference on Complex Networks and Their Applications* (pp. 231–243). Springer.

Poulin, V., & Théberge, F. (2019). Ensemble clustering for graphs: Comparisons and applications. *Applied Network Science, 4*(1), 51.

Qureshi, K. N., & Abdullah, A. H. (2013). A survey on intelligent transportation systems. *Middle-East Journal of Scientific Research, 15*(5), 629–642.

Rath, M., Pati, B., & Pattanayak, B. (2019, January). An overview on social networking: Design, issues, emerging trends, and security (pp. 21–47).

Shahraki, A., Taherkordi, A., Haugen, Ø., & Eliassen, F. (2020). Clustering objectives in wireless sensor networks: A survey and research direction analysis. *Computer Networks, 180*, 107376.

Soares, V., & Rodrigues, J. (2021, January). Vehicular delay-tolerant networks (pp. 59–78).

Sun, Z., Liu, Y., Wang, J., Anil, C., & Cao, D. (2020). Game theoretic approaches in vehicular networks: A survey. *arXiv preprint arXiv:2006.00992.*

Tai, C.-F., Chiang, T.-C., & Hou, T.-W. (2011). A virtual subnet scheme on clustering algorithms for mobile ad hoc networks. *Expert Systems with Applications, 38*(3), 2099–2109.

Vaidya, P. M. (1989). An O(n log n) algorithm for the all-nearest-neighbors problem. *Discrete & Computational Geometry, 4*(2), 101–115.

Yick, J., Mukherjee, B., & Ghosal, D. (2008, August). Wireless sensor network survey. *Computer Networks, 52*, 2292–2330.

Younis, O., & Fahmy, S. (2004). Distributed clustering in ad-hoc sensor networks: A hybrid, energy-efficient approach. In *IEEE INFOCOM 2004* (Vol. 1). IEEE.

Appendix

The Appendix contains the pseudocode of the algorithms that we are examining in the study.

Algorithm 1 Greedy algorithm

```
Let $clustering_{old}$ be singleton clustering
Let $eff_{old}$ be efficiency function for singleton clustering
while $criterion_{stop} = true$ do
  for $i = 1$ to $m$ do
    for $j = i+1$ to $m$ do
      Merge clusters $i, j$ into one cluster
      Calculate new efficiency function $eff_{new}$ for new clustering
      if $eff_{new} < eff_{old}$ then
        Let $clustering_{tmp}$ be clustering with merged $i, j$
        Let $eff_{tmp}$ be efficiency value for clustering with merged $i, j$
      end if
    end for
  end for
  Let $clustering_{old}$ be $clustering_{tmp}$
  Let $ef_{old}$ be $eff_{tmp}$
end while
```

Algorithm 2 Naive algorithm

```
Let $clustering_{old}$ be singleton clustering
Let $eff_{old}$ be efficiency function for singleton clustering
for $i = 1$ to $m$ do
    Let j be randomly picked cluster
    Merge clusters i, j into one cluster
    Calculate new efficiency function $eff_{new}$ for new clustering
    if $eff_{new} < eff_{old}$ then
      Let $clustering_{old}$ be clustering with merged $i, j$
      Let $ef_{old}$ be efficiency value for clustering with merged $i, j$
    end if
end for
```

Algorithm 3 Louvain algorithm

```
Let $clustering_{old}$ be singleton clustering
Let $eff_{old}$ be efficiency function for singleton clustering
while $criterion_{stop}$ = $true$ do
    for $i = 1$ to $n$ do
        for $j$ = neighbors(i) do
            $criterion_{stop}$ = $false$
            Change the assignment of $i$ to the cluster of $j$
            Calculate new efficiency function $eff_{new}$ for new clustering
            if $eff_{new} < eff_{old}$ then
                Let $clustering_{old}$ be the new clustering
                Let $eff_{old}$ be efficiency function value for the new clustering
                $criterion_{stop}$ = $true$
            end if
        end for
    end for
    for $i = 1$ to $clusters_{number}$ do
        $weights_{new}$[i,i] = sum of weights($G$) of edges from new cluster $i$
        for $j = i+1$ to $clusters_{number}$ do
            $weights_{new}$[i,j]=sum of weights($G$) of edges from cluster $i$ to $j$
            $weights_{new}$[j,i]=sum of weights($G$) of edges from cluster $i$ to $j$
            $G$ = graph($clustering_{oLd}$) with $weights_{new}$
        end for
    end for
end while
```

Algorithm 4 Stochastic hill climbing

```
Let $clustering_{old}$ be singleton clustering
Let $eff_{old}$ be efficiency function for singleton clustering
for $i = 1$ to $m$ do
    Let $sampled$ be the set of sampled clusters in proportion of $fraction$
    for j in $sampled$ do
        Merge clusters $i, j$ into one cluster
        Calculate new efficiency function $eff_{new}$ for new clustering
        if $eff_{new} < eff_{old}$ then
            Let $clustering_{old}$ be clustering with merged $i, j$
            Let $eff_{old}$ be efficiency value for clustering with merged $i, j$
        end if
    end for
end for
```

Algorithm 5 Simulated annealing algorithm

```
Let $clustering_{old}$ be singleton clustering
Let $eff_{old}$ be efficiency function for singleton clustering
Let $temperature = initial_{temp} / iteration_{no}$
for $i = 1$ to $m$ do
    Let $sampled$ be the set of sampled clusters in proportion of $fraction$
    for j in $sampled$ do
        Merge clusters $i, j$ into one cluster
        Calculate new efficiency function $eff_{new}$ for new clustering
        if $eff_{new} < eff_{old}$ then
            Let $clustering_{old}$ be clustering with merged $i, j$
            Let $eff_{old}$ be efficiency value for clustering with merged $i, j$
        else
            Let $criterion = \exp(-(eff_{new} - eff_{best})/temperature)$
            if $random_{vaLue} < criterion$ then
                Let $eff_{old}$ be efficiency value for clustering with merged $i, j$
            end if
        end if
    end for
end for
```