Optimisation of Car Electronics

J. Binysh*, T. Francis* and G. Politis*
* *University of Warwick, Complexity Department*

***Abstract*—Car electronics networks consist of Electronic Control Units (ECUs), which communicate with each other via an electronic network. Jaguar Landrover (JLR) is interested in techniques which can suggest intelligent architectures for this network, a problem they refer to as the N^2 *Problem*. In this report, we propose modelling this electronic network as a weighted, directed network (graph), with edge weights representing a measure of similarity between ECUs. We then discuss the application of community detection algorithms to this network as a method of solving the N^2 problem, and compare several recent algorithms. Finally, we construct a connectivity network from a dataset provided by JLR, and apply these algorithms to it - we detect several community structures, and make suggestions for how these results might influence future network designs.**

I. Introduction

Car electronics networks consist of *Electronic Control Units* (ECUs), each responsible for its own set of low level functions. These ECUs are connected together in a network architecture, and communicate through it. Typically, a network will be broken into several sub-networks, with communication between them occurring via a Gateway (Fig. 1). Implementing high level functionality in a car will require ECUs to co-ordinate the use of their functions through this network.

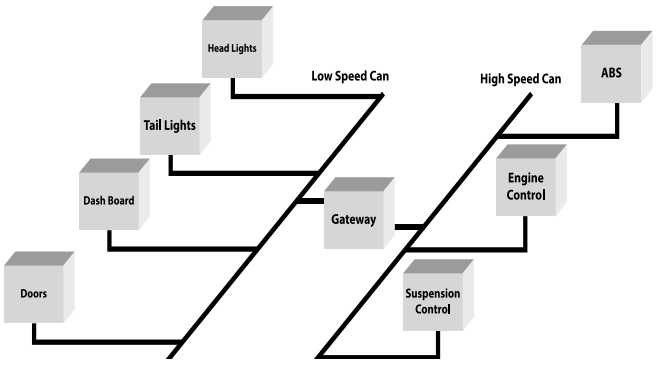


Fig. 1. An example of a car electonics network architecture. Two data buses, with several ECU's connected to each bus, and a gateway ECU connecting the two buses.

Jaguar Landrover (JLR) are interested in techniques which will suggest 'optimal' network architectures, a problem they refer to as the N^2 *problem* [1]. Currently, architectures are designed using 'best practice' [2], often constrained by legacy considerations. As JLR seek a major redesign of their network architecture, there is scope for tools which will aid engineers by automatically suggesting architectures. In this report, we will survey techniques which can be used to make such suggestions, as well as discussing and demonstrating their applicability to a car electronics network.

The report is structured as follows: In section II we will discuss in detail the design of car electronics architectures, before suggesting how they might be modelled as weighted networks[1], and discussing which properties of this network could be optimised. In sections III and IV, we introduce a technique, *community detection*, which can be used to perform this optimisation, before discussing a variety of algorithms which perform it, and how they might be assessed. In section V, we will briefly discuss related methods which may be of use to JLR. In section VI, we will demonstrate the application of these techniques to a dataset provided by JLR, before suggesting further work and concluding in section VII.

II. Car Electronics networks

Car electronics networks consist of Electronic Control Units (ECUs), which communicate via an electronic architecture. Sharing data amongst ECUs removes the need for redundant sensors, and allows complex functionality, distributed over many ECUs, to be implemented.

Each ECU knows which messages it sends to, and receives from, other ECUs. A message sent across the network may be thought of as having a type, a single sender ECU, a list of receiver ECUs, and some content. So, separate from the electronic architecture, there is a network of connectivity between ECUs, which any architecture will need to implement.

A. The Current Architecture and its Limitations

As it stands, architectures which allow this information sharing typically consist of several long data buses, with many ECUs connected to them, through which information is sent and received (Fig. 2).

[1]We mean network in the mathematical sense of a graph [3], rather than in any engineering sense.

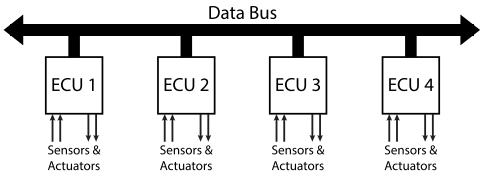


Fig. 2. ECUs control low level functions, and communicate with one another via a bus.

When a new ECU is to be added to the architecture, it is attached to the most appropriate bus, as judged by an engineer. A typical network may consist of approximately 100 ECUs, each responsible for perhaps several hundred low level functions. This architecture reportedly presents several problems:

1) If an ECU fails, it is not easy to localise the error - every ECU on a bus must be tested.
2) Having all messages passed along a single bus brings trouble with bus message capacity the more ECUs are added.
3) A bus cannot accommodate an indefinite number of ECUs - it is limited to around 30.

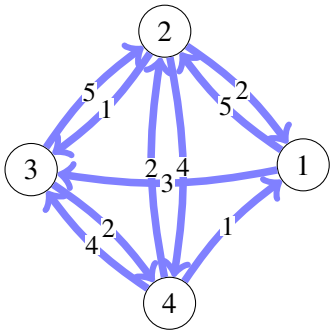
B. Future Architecture Designs

Because of these limitations, future designs will split ECUs up into domains [4]. Within a domain, ECUs will be connected to one another, and there will be sparse inter-domain connections. It is hoped that this design will make it easier to localise errors and minimise inter-domain network talk. The question then becomes, how should the division be done? One approach [2] [4] is to specify broad classes of functionality into which ECUs can be assigned by hand. However, there is no guarantee that an ECU has one clearly defined type of functionality - an ECU may have functions which are used in entertainment, safety, and many other types of functionality. If obvious divisions of ECUs cannot be made, then there is no guarantee that, when they are partitioned into domains, these domains will achieve the goals stated above. How then should we go about detecting these domains?

C. Our Model

We model the ECUs as nodes in a network [3], and use the terms interchangeably. The edges represent messages sent and received between ECUs. They are directed, representing the fact that messages are sent from one ECU to another, and weighted.

A network can be represented by its *adjacency matrix*, a matrix A where an element A_{ij} is non-zero if an edge exists from node i to node j, with its value being the weight of that edge. Note that, since edges are directed, in general A is not symmetric. A simple network, and its adjacency matrix, are shown in Fig. 3.



$$A = \begin{pmatrix} 0 & 5 & 3 & 0 \\ 2 & 0 & 1 & 4 \\ 0 & 5 & 0 & 2 \\ 1 & 2 & 4 & 0 \end{pmatrix}$$

Fig. 3. A simple network and its adjacency matrix. The weights shown on the network links appear in the adjacency matrix, with a zero meaning no link.

We propose that the weight of an edge between two nodes should represent a measure of similarity between two ECUs, and should be used to judge whether the ECUs should reside in the same domain. This weight could represent several related quantities, depending on what is considered an appropriate measure of similarity. Some examples are:

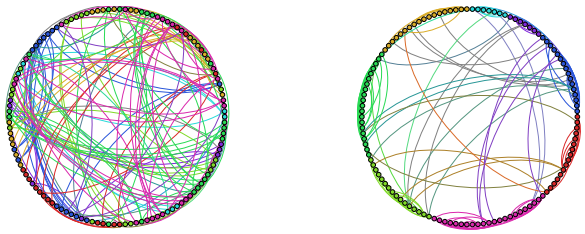
1) ***Number of messages shared between two ECUs***: The weight could simply be the number of messages one ECU sends to another.
2) ***Total information content***: If the number of bits of data in a message can vary between message types, we could multiply the number of messages shared between ECU's by the number of bits of data each message contains.
3) ***Expected frequency of communication***: One ECU may only send a single message to another ECU, but during car operation it may do so very frequently. We could weight each message by how frequently it is sent.
4) ***Signal Latency***: Messages sent over the network have expiry times - they must complete their journey before this time elapses, or they expire. It would make sense to weight messages using their expiry times, such that ECU's with low expiry time communications are placed in the same domain.

One could imagine constructing some combination of these factors. For example, the weight of an edge from node i to node j could be

$$W_{ij} = \sum_k (f_{ij})_k (s_{ij})_k \tag{1}$$

where the sum is taken over all messages sent between the two ECUs, with $(f_{ij})_k$ the frequency of the k^{th} message between nodes i and j, and $(s_{ij})_k$ its size.

Once this weighted connectivity network is constructed, it needs to be examined to see if there are natural groupings of nodes into domains within it. This can be accomplished using *community detection* algorithms[2]. These algorithms find natural groupings of nodes in a network, based on the topology of the network, and the weights of the edges. Fig. 4 demonstrates an example of their use.



(a) Randomly ordered nodes

(b) Sorted using Community detection

Fig. 4. A network derived from an electric circuit [5]. The nodes are circuit elements, and the edges wires between them. 4a has the nodes randomly ordered in a circle. 4b shows them sorted according to natural groupings of nodes, as found by a community detection algorithm. The colours represent detected communities. Note the relative sparsity of inter-community links.

There is a large literature on this subject [6] - in the following sections (III, IV) we give an outline of recent theory, and discuss several notable algorithms, all available under the GNU General Public License, which JLR may wish to use.

III. Community Detection Theory

In this section, we will consider a network $G = (V, E)$, with $|V| = N$ vertices, $|E| = m$ edges and adjacency matrix A. We will denote in-degree and out-degree by k^{in} and k^{out}. If we are discussing an undirected network, the degree is simply denoted by k.

Intuitively, a community is a group of nodes within a network with a relatively high density of links between them compared to links to nodes outside of the community (Fig. 5) - any precise definition of a community should respect this intuition.

[2]Note that a *domain*, in the above engineering language, is a *community* in the community detection literature. We will use these terms interchangeably.

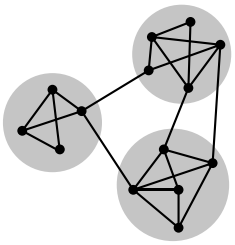


Fig. 5. A network with an intuitively clear community structure, which is captured by the partition chosen, shown in grey. Image reproduced from [7].

However, there is no agreed upon definition of a community - many different ones have been proposed, all compatible with the above intuitive description, and the most useful definition may vary depending on the problem at hand. To give some intuition for typical definitions in the literature, here we present two examples:

1) ***Intra-cluster link density***: [6] proposes a fitness function as follows. Given a group of nodes $U \subset V$, we can define the fraction of possible edges which are present between these nodes

$$\delta_{int}(U) = \frac{\#\text{internal edges of } U}{\frac{1}{2}|U|(|U|-1)} \tag{2}$$

and require that it exceeds some threshold for it to be considered a community.

2) ***Weak and strong communities***: [8] defines a *strong community* as a group of nodes in which each node has more links to other nodes within the group, than to nodes outside. The authors define a *weak community* as a group of nodes which has, in total, more links within it than to the nodes outside it.

Both of these definitions are *local* - they only depend on the nodes inside the community. By contrast, *global* definitions quantitatively measure the amount of community structure in the network as a whole. As input, these definitions take a network, and a *partition* of that network.

Given a set of community labels C, with $|C| = n_C$, a partition $\mathcal{C}$ is a map $f : V \to C$ of every node $i \in V$ to a community $c \in C$. We denote the labelling of node i into community c as c_i. Fig. 5 shows an example partition - for any network there are many possible ones.

Given a partition, as output global measures give a score indicating the quality of the community structure in the given partition. These measures make sense if there is reason to believe a global community structure exists in the network - rather than communities simply existing within the network, the network as a whole possesses a modular structure.

In the following subsections we will describe two well known global measures, the *modularity*, Q, and the *minimum description length*, L.

A. The Modularity

The underlying idea of the modularity is to compare the number of edges within each community in $\mathcal{C}$ to how many we expect to find in a null model with no community structure [9]. We will begin by discussing the undirected, unweighted case. In the most common null model [3], networks are generated by placing links between nodes i and j independently with probability

$$\frac{k_i}{2m} \times \frac{k_j}{2m} \tag{3}$$

where k_i and k_j are the degrees of the nodes i and j in the original network. Under this model, the expected number of links between nodes i and j is

$$\frac{k_i k_j}{2m} \tag{4}$$

and so the expected number of intra-community edges is

$$\frac{1}{2}\sum_{ij} \frac{k_i k_j}{2m}\delta(c_i, c_j) \tag{5}$$

where the Kronecker delta $\delta(c_i, c_j)$ ensures we are only counting edges within a given community. Note that this allows multiple edges between two nodes. For the actual given network, the total number of intra-community edges is:

$$\frac{1}{2}\sum_{ij} A_{ij}\delta(c_i, c_j) \tag{6}$$

Subtracting Eq. (5) from Eq. (6), and normalising by m gives the modularity, Q, of the partition $\mathcal{C}$:

$$Q_{\mathcal{C}} = \frac{1}{2m}\sum_{ij}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\delta(c_i, c_j) \tag{7}$$

The modularity can be extended to weighted, directed networks [10] [11]. Suppose A is now a weighted, directed matrix. We define the *in-degree* of node i as

$$k_i^{\text{in}} = \sum_j A_{ij} \tag{8}$$

and the *out-degree* as:

$$k_j^{\text{out}} = \sum_i A_{ij} \tag{9}$$

These are the total weights flowing in and out of a node. The modularity is then extended in a natural way:

$$Q_{\mathcal{C}} = \frac{1}{m}\sum_{ij}\left(A_{ij} - \frac{k_i^{\text{in}} k_j^{\text{out}}}{m}\right)\delta(c_i, c_j) \tag{10}$$

A positive Q tells us we have more links than expected and is indicative of community structure. The measure must be used with care - it is not equal to 1 even for networks with links only within communities [3]; unnormalized, it cannot be used to compare networks of different sizes [6]; and it is well known that it suffers from a *resolution limit*, where it can fail to detect communities which are small relative to the network size [12]. Nevertheless, one common strategy for community detection is to search for the partition of G with maximum modularity.

We decided to present in some detail the modularity, because it gives intuition for the sorts of measures commonly used to assess the community structure of a partition. It is also the basis of a large class of algorithms, some of which we will discuss later. However, many other measures exist [6], and very often, rather than explicitly defining communities, an algorithm will simply present a method to be carried out on a network, the output of which is the communities. One recent, and notable, alternate measure is the *minimum description length* [13].

B. Minimum Description Length

The idea behind this measure is that, when encoding a description of a network, community structure can be exploited to give efficient encodings. [13] considers the problem of encoding a random walkers trajectory on a weighted, directed network using a two-stage Huffman encoding scheme. At the top level, each community is given a unique name, and then names are re-used within communities. Given a partition $\mathcal{C}$, the *map equation*

$$L(\mathcal{C}) = r_{\curvearrowright} H(R) + \sum_{i=1}^{m} p^i_{\circlearrowright} H(P^i) \tag{11}$$

tells us the average length of codeword needed to encode a single step in the path of a random walker on the network.

The first term, $H(R)$, is the Shannon Entropy of the random variable R. The states of R are community labels, and its distribution $r_{i\curvearrowright}$ gives the probability a random walker will leave community i. These quantities can easily be computed using the theory of Markov Chains. Then

$$r_{\curvearrowright} = \sum_i r_{i\curvearrowright} \tag{12}$$

and we see the first term of Eq. (11) is interpreted as the average length of codeword needed to describe movement between modules. $H(P^i)$ is the Shannon Entropy of the random variable P^i. The states of P^i are labels for nodes within the i^{th} community (these labels are re-used between communities). The second term is interpreted as the length of codeword needed to describe movement within modules.

$L(\mathcal{C})$ is dependent on the partition $\mathcal{C}$ which is used, and a short $L(\mathcal{C})$ indicates a partition which has captured the community structure present in the network. Another class of algorithms thus use $L(\mathcal{C})$ as a quality function to optimise.

We have now seen some examples of the sorts of structures one might wish to detect, or quantities one might wish to optimise, to go about detecting communities. In the following section we discuss how algorithms can use these ideas to perform community detection.

IV. Community Detection Algorithms

As we understand it, JLR's current approach to the problem stated in section II is to write down the adjacency matrix of the network, and exhaustively try row and column permutations to bring it into a form where groupings can be read off. It is reported that their method breaks down once the number of nodes in the network exceeds 30.

Two problems arise with this method. The first has been addressed above: how should we 'read off' groupings? In other words, what criteria do we use to assess communities. The second problem is: given a criteria to determine whether we have communities or not, how should we search through solution space?

The number of possible permutations of a matrix rises factorially with its size. Exhaustive searches of permutation space therefore quickly become computationally intractable as network size increases, explaining why, above 30 nodes, JLR's current method breaks down. This is not a problem unique to their method - the number of partitions, as defined above, of a network rises super-exponentially with the network size [6]. So an exhaustive search of partition space, for example attempting to maximise Q, will not be possible. Indeed, recently modularity maximisation was proved to be an $\mathcal{NP}$ - hard problem [6], as is searching for both types of communities defined in section III [6]. To deal with these issues, algorithms typically rely on heuristic search methods.

In this section we will present three recent algorithms, representing the 'state of the art', before discussing how the performance of these algorithms, and indeed any community detection algorithm, might be assessed. Justification for why we have chosen these algorithms in particular will come in subsection IV-B.

A. Recent Community Detection Algorithms

The three algorithms discussed below all work by optimising a quality measure - either the modularity, Eqs. (7, 10), or the minimum description length Eq. (11) - by using a heuristic search procedure through partition space.

1) ***The Louvain method*** [14]
This algorithm optimises the modularity Q with a greedy search through partition space. It begins with each node as its own community. It then considers each node in turn, testing whether Q can be increased by moving the node to a community it is connected to, and doing so if Q increases. After this, it takes the new communities formed, considers them single nodes in a new hyper-network, and iterates this procedure until no increase in Q can be found. This produces a hierarchy of communities, of increasing Q. The method is experimentally found to be $O(N)$. In implementations currently available, it takes into account link weights, but not directions.

2) ***Infomap*** [13]
This algorithm was introduced in [13] along with the minimum description length Eq. (11), and attempts to optimise it. Implementations have varied over time; the current one uses a greedy search procedure similar to the Louvain method.

3) ***Combo*** [15]
This algorithm can use either the modularity or the minimum description length as its objective function. It searches partition space as follows: Given a partition $\mathcal{C}$, for each community $c \in C$, redistributions of every node in c into other communities are trialled. The trial with the largest increase in the objective function is accepted.
The redistribution of nodes from the source community into other (possibly empty) communities is fashioned after the *Kernighan-Lin* algorithm [16]. The algorithm starts with a list of all nodes in the source community. The node which gives the highest gain (or smallest loss) in the objective function when moved to another community is then found. This node is moved (and cannot be considered again), and the value of the objective function is noted. The process is then iterated for all nodes in

the source community. When all nodes have been considered, the intermediate state with the largest value of the objective function is accepted. It is reported that the upper bound of the execution time scales as $O(N^2 \log n_C)$.

B. Assessing the Performance of Algorithms

When assessing how an algorithm performs, two factors are relevant: How execution time varies with network size, and how accurately an algorithm uncovers the 'true' community structure within a network. On the first point, it should be noted that all modern algorithms can evaluate a network of thousands of nodes in seconds or minutes on a typical laptop, depending on the algorithm. Indeed, greedy algorithms have been applied to networks of several million nodes. So for an ECU network of 100 nodes, it will not be speed, but quality of results, which will primarily determine the choice of algorithm.

Assessing the quality of results from a community detection algorithm is not straightforward. We have seen that there are many different definitions of community, and several commonly used objective functions which are considered proxies for community structure, each with some theoretical justification behind them. To be confident in any method we would like to see it uncover community structure we 'know' to be present in a network; and in the absence of community structure, for example in a random network, we would expect to see it give a null result.

One method of assessment is to use existing datasets with built in community structure separate from the network topology, which we wish to rediscover using the algorithm. Another is to use synthetic data generated with predefined community structure. This is the approach taken in the recent *LFR benchmark* [17] [18] [19], a tool which generates networks, with built in community structure, which show properties mirroring those found in many real world networks - power law distributions of node degree and community size. This benchmark improves on previous network generation tools to present a standardised set of challenging networks which algorithms can be tested on.

We can measure the extent to which the community structure found by our algorithm matches that built in to the data using the *normalised mutual information* (NMI) [20], a measure drawn from information theory. In this measure, given two partitions $\mathcal{X}$, $\mathcal{Y}$, we consider each pair of community labellings (x_i, y_i) of node i as a realisation of a random variable pair (X, Y). We then compute the mutual information between these two random variables, and normalise it:

$$I(X,Y) = \frac{2\sum_x \sum_y P_{X,Y}(x,y) \log \frac{P_{X,Y}(x,y)}{P_X(x)P_Y(y)}}{\sum_x P_X(x) \log P_X(x) + \sum_y P_Y(y) \log P_Y(y)} \tag{13}$$

We approximate

$$P_{X,Y}(x,y) = \frac{n_{xy}}{n} \tag{14}$$

where n_{xy} denotes the number of nodes found in both community $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. Similar approximations are made for the marginal probabilities. $I(X,Y) \in [0,1]$, and, intuitively, tells us how much information we learn about one partition, given the other - if they are identical, $I(X,Y) = 1$, whereas if they are uncorrelated, $I(X,Y) = 0$.

Using the LFR benchmark to generate networks, and the NMI to assess how well algorithms capture their community structure, [19] compares a range of algorithms, concluding that Infomap, followed by the Louvain method perform best on the tests set. This justifies our choice of these particular algorithms for discussion.

Combo was published after this comparative analysis was done, and the authors report it capable of finding partitions of better modularity than all other algorithms available at the time of publication (2014). They further report that it performs comparably to Infomap on minimum description length optimisation. These are promising results - however, as discussed above, assessing the quality of algorithms is problematic, and it is not clear that these findings will translate to good performance on the benchmark found in [19], which is a better measure of the quality of an algorithm. To be able to assess the value of Combo, it should be submitted to the same tests as [19].

The LFR benchmark can generate (un)weighted, (un)directed networks to be tested. To begin, we compared Combo[3], Infomap, and the Louvain method on unweighted, undirected networks (Fig. 6). In all that follows, we use the same parameters as those found in [19], namely: The network size was set to 1000 nodes, the average degree was set to 20, the maximum degree was set to 50, the exponent for the degree sequence was set to 2, the exponent for the community size distribution was set to 1, and the community sizes were set to a range from 10 to 50. Each data point is averaged over 100 generated networks, with errors shown.

In Fig. 6, the algorithms were run for a range of values of the *topological mixing parameter* $0 \leq \mu_t \leq 1$. This

[3]Note that Combo can use modularity or minimum description length as its quality function. Currently, the only publicly available implementation uses modularity, and this is the version we use.

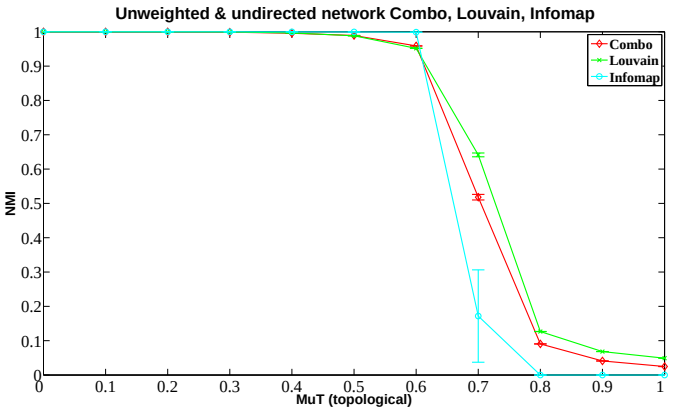


Fig. 6. A comparison of the normalised mutual information for Combo, the Louvain method and Infomap on undirected, unweighted networks generated using the LFR benchmark [17]. Each generated network has $N = 1000$, and each data point is averaged over 100 realisations. LFR parameter values can be found in section IV-B.

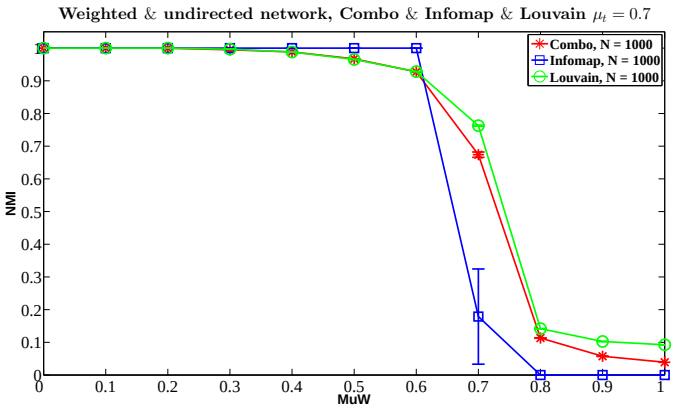


Fig. 7. A comparison of the normalised mutual information for Combo, the Louvain method and Infomap on weighted, undirected networks, for $\mu_t = 0.7$.

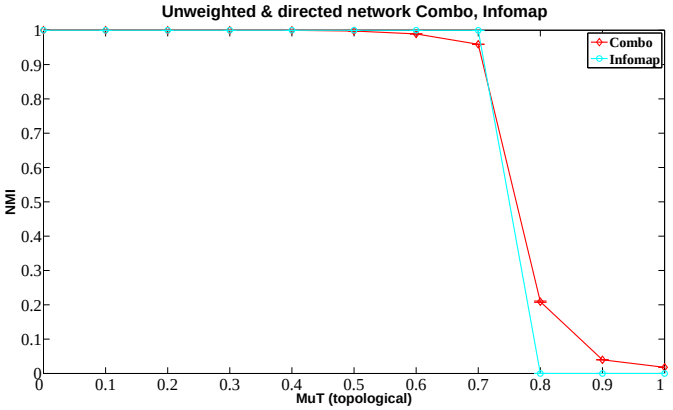


Fig. 8. A comparison of the normalised mutual information for Combo and Infomap for unweighted, directed networks.

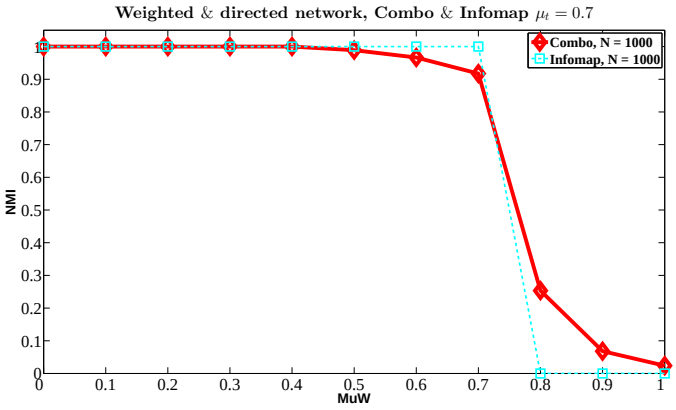


Fig. 9. A comparison of the normalised mutual information for Combo and Infomap on weighted, directed networks, for $\mu_t = 0.7$.

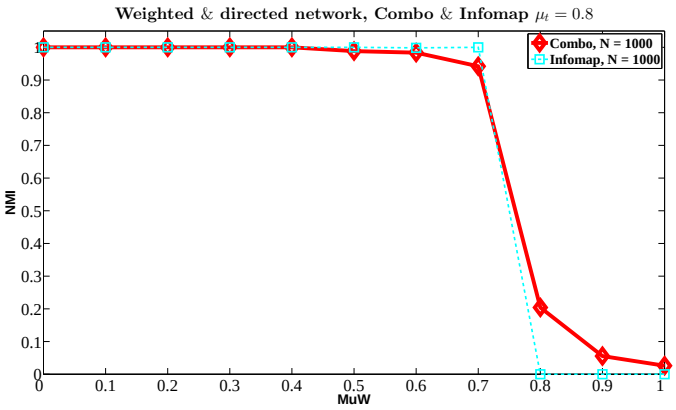


Fig. 10. A comparison of the normalised mutual information for Combo and Infomap on weighted, directed networks, for $\mu_t = 0.8$.

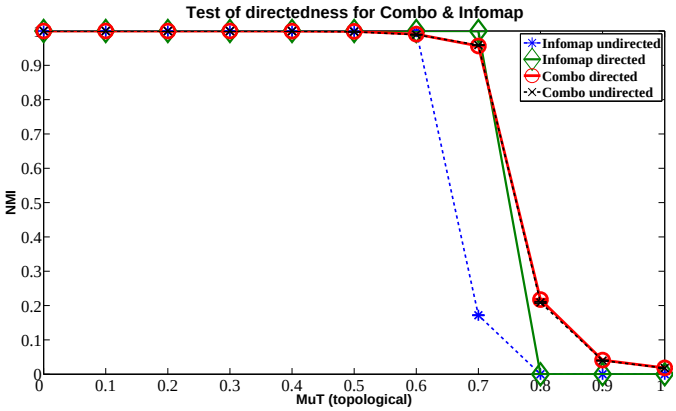


Fig. 11. Tests on the importance of edge direction for community detection on Combo and Infomap. A directed, unweighted graph is generated, and the algorithm is run on it. Edge direction is then neglected, and the algorithm run again.

parameter is defined as the ratio between the external degree of a node i (the number of links it has to nodes outside its own community) and the total degree of the node. In the LFR benchmark, μ_t is the same for all nodes, and measures the strength of the community structure embedded in the network - [17] shows that, for the networks we consider, communities are in principle detectable until $\mu_t = 0.95$.

The LFR benchmark can also generate weighted networks. After generating an unweighted network, the benchmark uses a weight mixing parameter, μ_w, to assign weights to the edges. μ_w is the ratio between the external strength of a node (the sum of its edge weights) and all other nodes in its community. To investigate the effect of edge weights on algorithm performance, in Fig. 7 we fix $\mu_t = 0.7$, near the transition in Fig. 6, and vary $0 \leq \mu_w \leq 1$. We find similar results to the unweighted, undirected case, with the Louvain Method slightly outperforming Combo, and Infomap outperforming both until $\mu_w = 0.6$.

We can also assess the algorithms performance on directed graphs, shown in Figs. 8, 9 and 10 - current implementations of the Louvain method do not take edge direction into account, so we only compare Combo and Infomap. We again find Infomap outperforming Combo for smaller values of μ_t and μ_w, with its performance rapidly dropping off for larger values.

Theoretically, when we neglect edge direction, we are not using the full information present in the network topology - taking direction into account should lead to better community detection. Further, direction of edges is an important part of how we model JLR's ECU networks. Thus, it is interesting to test whether neglecting edge direction really does affect the quality of community detection. To test this, we generated directed, unweighted networks, and ran Combo and Infomap on them. We then neglected edge direction, and ran the algorithms again on the undirected networks. The results are shown in Fig. 11. We find that neglecting edge direction does not significantly affect the performance of Combo, though it does degrade Infomap's results.

Taken together, these findings do not suggest a clearly superior algorithm. The Louvain Method outperforms Combo on modularity optimisation, but does not take into account edge direction. Infomap fares slightly better than Combo in some parameter ranges, but worse in others. We also find edge direction to be unimportant for community detection on the LFR benchmark graphs, though reports of neglecting edge direction leading to incorrect community detection do exist in the literature [10]. We will discuss this last point further in section VII.

V. Other Techniques

In this section, we will briefly survey other techniques, related to those above, which may be of use to JLR.

A. Optimisation Functions

[1] suggests a method to simplify designs where, given an adjacency matrix (an *N squared chart* or *design structure matrix* in the language of the paper), a genetic algorithm can be used to suggest permutations of the rows and columns, which are assessed using some cost function. This is a very general setup - the use of a genetic algorithm is not necessary; any stochastic optimisation method, for example Simulated Annealing [21], could be used. Further, while [1] presents a weighted average of the distances of elements in the matrix from the diagonal as the cost function, in principle many other functions could be used, depending on what the designer is interested in optimising.

1) ***Matrix bandwidth:*** One commonly optimised quantity is the matrix bandwidth. Given a matrix A, the bandwidth $B(A)$ is defined as:

$$B(A) = \max_{i,j}\{|i-j| : \ A_{ij} \neq 0\} \tag{15}$$

In words, the largest gap between a non-zero element and the matrix diagonal [22] [23]. Motivation for optimising this quantity comes in several forms. As in [1], one is to simplify the N^2 chart for the designer, allowing them to restrict their attention to a sub band of the full matrix. Another comes from electronic network design, where the bandwidth is interpreted as the maximum signal delay time in a circuit.

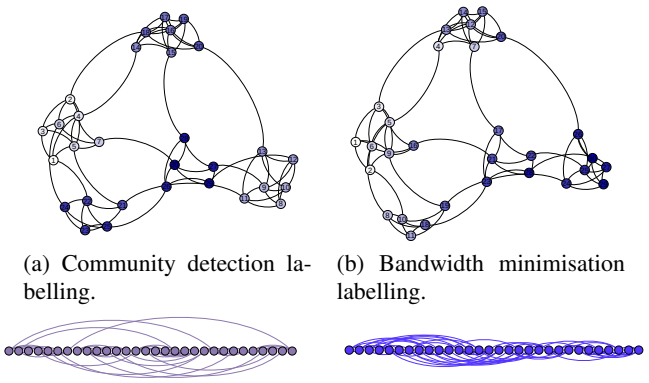
2) ***Weighted averages:*** As in [1], one may wish to optimise some form of weighted average

$$B(A) = \sum_{i,j} A_{ij}|i-j|^{\alpha} \tag{16}$$

for $\alpha > 0$. This form accounts for matrix element weights, with a value of α which can be tailored to the problem.

It should be noted that, though these techniques seem similar to community detection, they may give very different results. [1] presents the example of weighted averages as an optimisation function, and gives an example matrix, before and after this analysis ([1], Figs. 3 and 5). However it is not clear the methods suggested in [1] will yield the adjacency matrix it desires. To demonstrate this, we take an example matrix with strong community structure, and apply a community detection algorithm

(the Louvain Method)[4] and a *bandwidth minimisation* algorithm (the Cuthill-Mckee algorithm [22]) to it. The results are shown in Figs. 12, 13.



(a) Community detection labelling.

(b) Bandwidth minimisation labelling.

(c) Community detection labelling, linear layout.

(d) Bandwidth minimisation labelling.

Fig. 12. 12a, 12b show labellings of nodes derived from a community detection algorithm, and those of a bandwidth minimisation algorithm, for a network of 30 nodes, Q = 0.85. Colour increases linearly with node label. 12c, 12d show these labellings linearly ordered. Note the difference in link distribution.

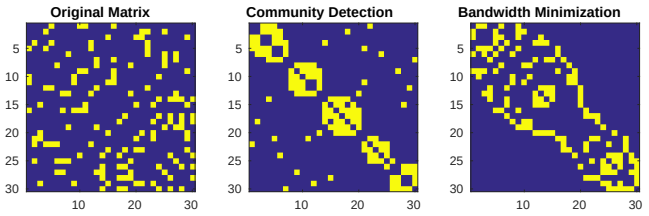


Fig. 13. The adjacency matrix of Fig. 12, permuted randomly, into communities, and into minimum bandwidth order.

We see that the results are very different - the matrix bandwidth fails to capture community structure within the graph entirely. This makes sense - there is no justification for why matrix bandwidth, or some variant of it, should measure community structure. Simply constructing an optimisation function which punishes elements far from the matrix diagonal will not lead to the identification of useful community structure, contrary to what [1] states. By contrast, the Louvain method captures the (intuitively obvious from Fig. 12a) community structure present.

B. Graph Partitioning

The *graph partitioning* problem [3] is defined as follows: Given a graph, divide its nodes into two separate groups such that the number of edges between the two groups is minimised. Importantly, the number of nodes in each group is fixed, in contrast to the community detection problem, which imposes no fixed size on the communities. Typically, division into more than two groups is achieved by recursively splitting the existing groups.

For JLR, the interest in this problem, and where it differs from the community detection approach, is in the predefined community size and number. It may be that external constraints force designers to put ECUs into a certain number of fixed size groups, but which ECUs go in which groups is to be optimised.

The problem suffers from the same super-exponential growth of possible divisions as community detection, and algorithms used to solve it are again heuristic. Two commonly used ones are *spectral partitioning*, an $O(N^2)$ method which works with the eigenvectors of the graph Laplacian [3] [24], and the *Kernighan-Lin* method [16], an $O(N^3)$ algorithm which repeatedly tries node swaps between the two groups, recomputing the number of intergroup edges each time. Combined, these methods can solve the graph partitioning problem for hundreds of thousands of nodes, well in excess of the typical number of ECUs JLR is interested in.

VI. Application to JLR Dataset

In this section, we present a dataset provided by JLR, and the results of applying the above techniques to it.

A. ECU network communication dataset

The dataset consists of a complete list of messages sent between ECUs, on two sub-networks of a typical car electronics system. The datasets are made up of 15 (subnetwork 1) and 14 (subnetwork 2) ECUs respectively. Each entry is of the form *(Message ID, Sender ECU, Reciever ECUs)*. The two subnetworks are connected together via a gateway node, which also has other subnetworks attached to it, which we do not have access to in the dataset. The setup is similar to that found in Fig. 1, except for the fact that the gateway ECU has several other subnetworks connected to it.

The dataset does not contain information on the frequency at which messages are sent, or their acceptable travel times, though these data are in principle available. As such, we simply weight edges by the number of messages sent from one ECU to another.

B. Results

Fig. 14 shows the network of connectivity constructed from the dataset. The central node is a Gateway ECU,

[4]Community detection algorithms don't offer a complete labelling of every node - the labels within communities are arbitrary. Here we have applied the Cuthill Mckee within each community to give a labelling.

labelled INST - messages sent to it are forwarded onto other ECU's, but the dataset does not contain their destinations (equally, messages sent from it originally came from another ECU, but we do not know which). They may be sent to ECU's outside of the subnetworks we have data for, or they may be communications between our two subnetworks - but we cannot infer it from the given dataset. Thus we cannot run the algorithms on the full dataset.

To account for this, we remove the gateway node, and all messages sent to and from it, from the dataset and focus on uncovering structure in the two isolated subnetworks.

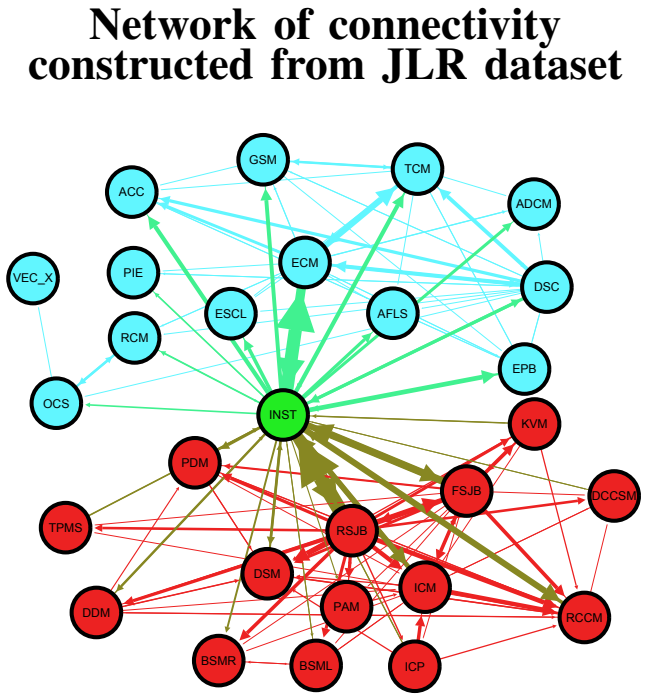


Fig. 14. Network of ECU connectivity constructed from JLR's electronic network dataset. The red (subnetwork 1) and blue (subnetwork 2) nodes show the sub-networks, with the gateway node, INST, shown in green. The thickness of the edges is defined by the average link weights between two nodes.

Figs. 15, 16 show the sub-networks with communities, as detected by Combo, shown in colour. It should be noted that, for networks this small, simply visualising them tells us a lot about the structure of communication. For example, inspection of Fig. 15 shows the majority of communication occurring between a few key nodes (ISM, RSJB, RCCM, FSJB and KVM), and shows that nodes DCCSM, PAM, BSML, BSMR and TPMS communicate strongly with RSJB and weakly with the rest of the network. This suggests isolating them from the main bus, and perhaps routing their communication with the rest of the network through the RSJB node. Fig. 16 shows a core of strong communication between the TCM, ECM and DCS nodes, with a periphery of nodes communicating with these three, and not between themselves. This sub-network structure suggests isolating the RCM, OCS and vec_x nodes.

The networks also demonstrate the importance of directed edges - for example, in Fig. 16 we see that communication with the ACC node is almost entirely one way.

On sub-network 1, Combo detects 4 communities, with a Modularity of 0.123. On sub-network 2, it detects 2 communities, with a Modularity of 0.154. As discussed above, positive Q is indicative of community structure. Here, the maximum. Figs. 17 and 18 show the results of the Louvain method applied to the same network. On sub-network 1, it detects 2 communities, with a modularity of 0.124. On sub-network 2, it detects 2 communities, with a modularity of 0.128[5]. The results reported by the Louvain method for the second subnetwork (Fig. 18) are similar to the ones of Combo. For the first sub-network, the Louvain method finds two communities - they are combinations of communities found by Combo (shown in green and light blue, purple and red in Fig. 15), with the exception of the PAM module. This merging of communities corresponds to a slight increase in modularity from Combo to the Louvain method - they are two related partitions of similar modularity. This is not unexpected - there may be many related partitions, all of similar modularities, near the true optimum [6].

Figs. 19 and 20 show the results of Infomap applied to the same network. The communities found on subnetwork 2 are very similar to those found with Combo, but there is substantial difference in sub-network 1, with all nodes except BSMR and BSML being placed in one community. We will remark further on this in the next section.

VII. Conclusions and Further Work

In this report, we have discussed methods capable of solving the N^2 problem. We have shown that, in contrast to JLR's current upper bound of 30 ECUs, techniques exist which can easily handle several thousand ECUs. We have discussed three examples, concluding that Infomap and Combo are two available algorithms capable of handling weighted and directed edges with excellent performance. We have also demonstrated how assessment of community detection algorithms may be

[5]For subnetwork 2, the different values of Q found from Combo vs. Louvain, despite the same partition being found, can be attributed to the fact one uses the weighted directed modularity, and the other only uses weights.

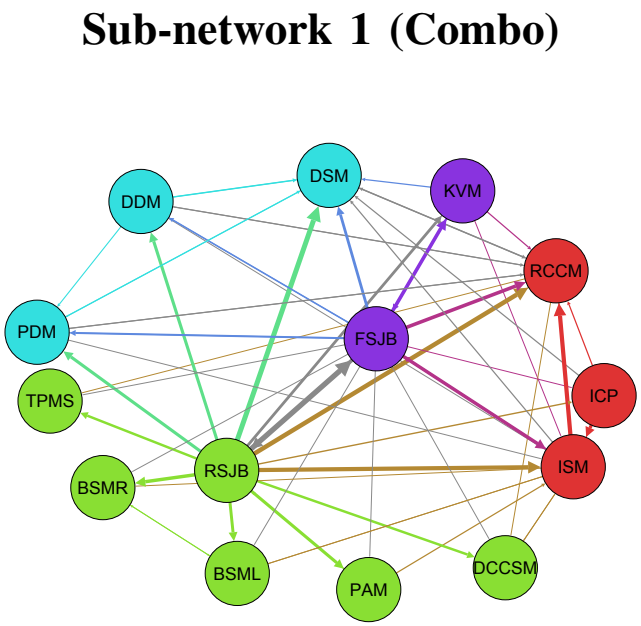


Fig. 15. A network of ECU connectivity, constructed from the first sub-network of the JLR dataset discussed in section VI. Each node is an ECU, and edges represent messages sent from one ECU to another. Communities, as detected by the community detection algorithm Combo, are shown as colours. Q = 0.123.

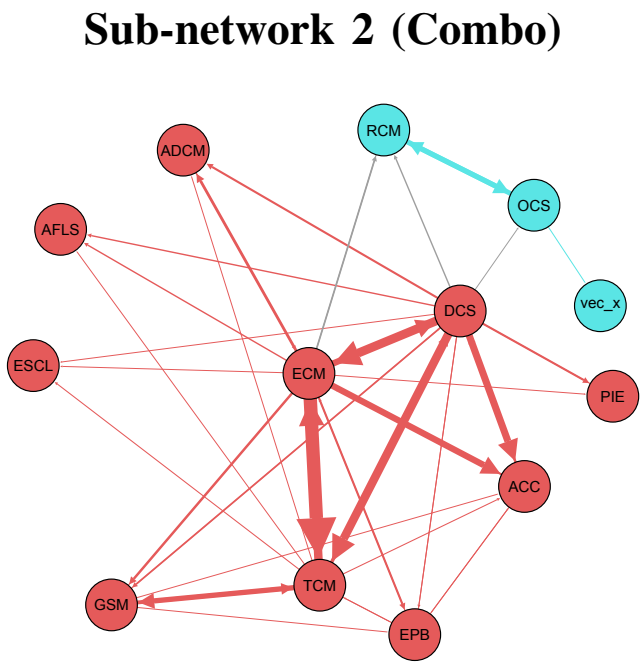


Fig. 16. ECU network constructed from second sub-network dataset, with communities, as detected by Combo, shown as colours. Q = 0.154.

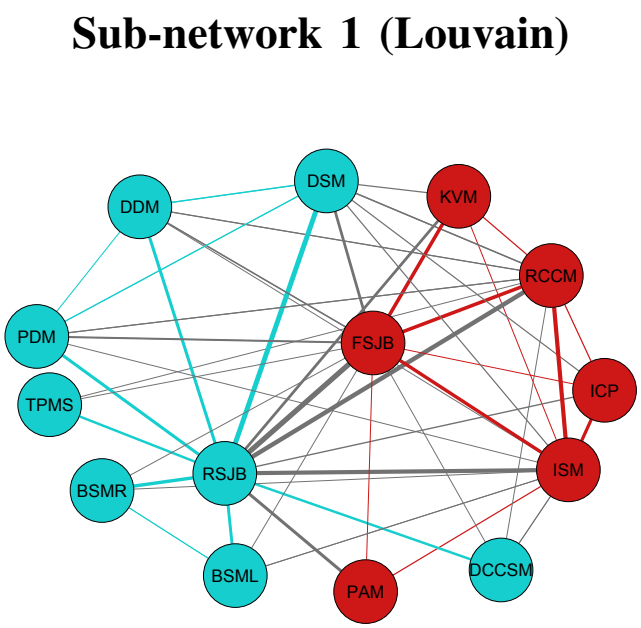


Fig. 17. ECU network constructed from first sub-network dataset, with communities, as detected by Louvain, shown as colours. Q = 0.124.

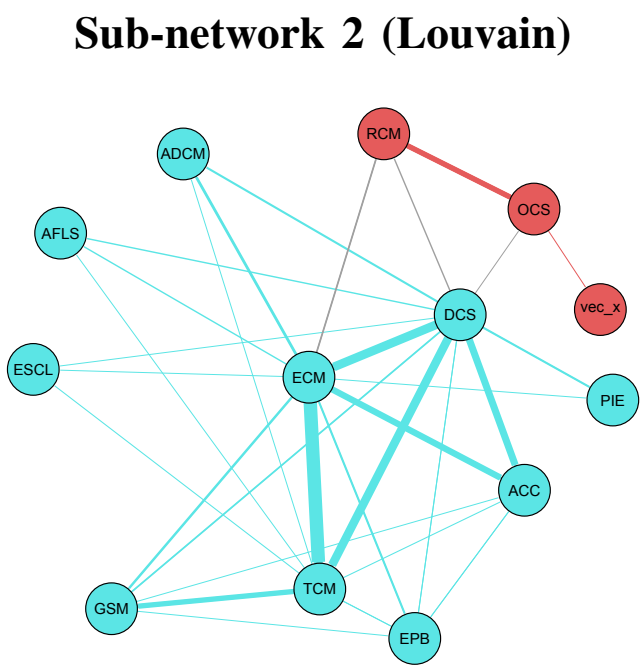


Fig. 18. ECU network constructed from second sub-network dataset, with communities, as detected by Louvain, shown as colours. Q = 0.128.

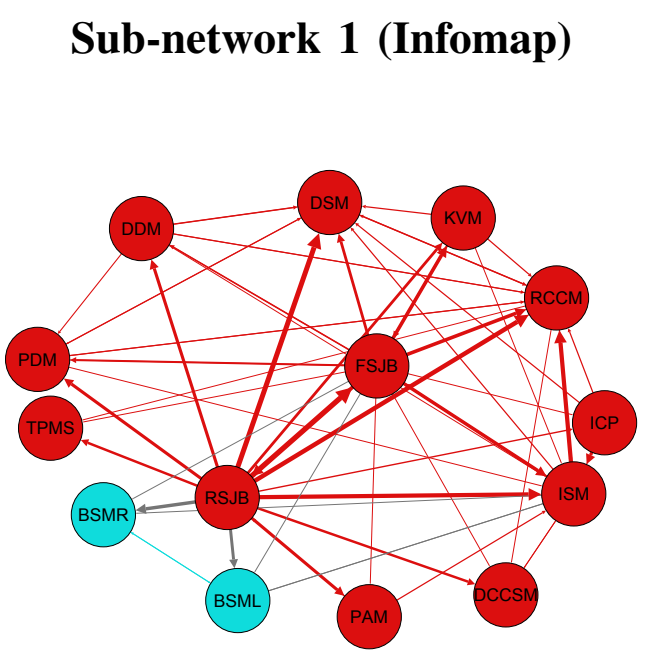


Fig. 19. A network of ECU connectivity, constructed from the first sub-network of the JLR dataset discussed in section VI. Each node is an ECU, and edges represent messages sent from one ECU to another. Communities, as detected by the community detection algorithm Infomap, are shown as colours.

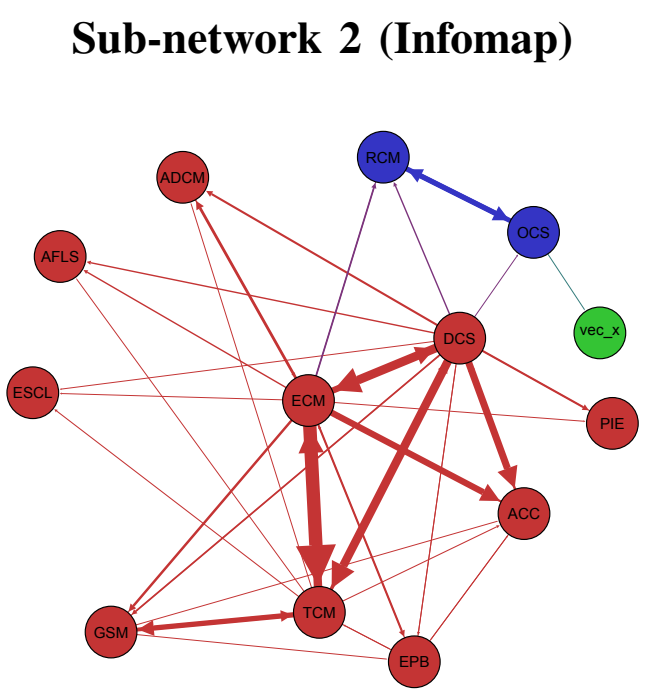


Fig. 20. ECU network constructed from second sub-network dataset, with communities, as detected by Infomap, shown as colours.

performed, and indeed what suitable definitions of a 'community' are. We have also briefly surveyed other, related, techniques which JLR may find a use for, and have demonstrated that finding the structures they desire is not as simple as [1] might suggest. Finally, we have applied these techniques to data provided by JLR.

It should be noted that, toward the end of this project, we discovered independent work [25] which discusses using community detection algorithms to improve in-vehicle security, and contains many ideas similar to those found in this report. The authors apply several community detection algorithms to ECU network data provided by Volvo. The dataset is weighted in a manner similar to ours, but is undirected. Their results show that using community detection algorithms can suggest designs which improve on those suggested by general guidelines.

One result which is similar to our findings is that when Infomap is applied to their dataset, it returns one community containing almost all nodes, which is not further commented upon. Future work could explore why this might be - [13] discusses the differences between the structures uncovered by modularity optimisation vs. code description length. It would be interesting to know what properties of these networks is causing Infomap to give these answers, given that Infomap has shown strong performance in [17] and our own LFR tests.

[25] also states that edge direction is unimportant when detecting communities in ECU connectivity networks. We have found mixed results on this point, with direction important for the results of Infomap, but not Combo when tested against the LFR benchmark, and Combo and the Louvain method (which neglects direction) giving similar but not identical results when applied to the JLR dataset. This is a further issue which could be explored when selecting appropriate algorithms for JLR to use.

We interpret the existence of [25] as validation of our approach, and suggest that there is certainly scope to apply the above methods to a JLR dataset, and reason to believe useful results will follow.

The results shown in section VI are a proof of principle - they demonstrate the applicability of the techniques discussed to datasets possessed by JLR. The obvious next step is to apply the method to a more complete dataset - one for the whole ECU network, rather than small subsections of it, and one which contains information

such as message frequency. [6]

It would also be of interest to discuss the results of community detection algorithms with JLR engineers, to see how groupings of ECUs suggested by the algorithms tie in to their knowledge of ECU functionality. It may even be interesting to compare the results of algorithms to architectures suggested by engineers.

Two additional considerations, known to be important to JLR, but not considered here, are message and node priority. It is known that ECU's are divided into different levels of importance - the most important being safety critical ECU's, for example - and it is likely these levels will need to be taken into consideration when making design suggestions. Exactly how is unclear - it may be that nodes of the same priority should preferentially belong in the same communities, which could be incorporated into edge weights, but more work is needed to clarify this. One suggestion would be to model these levels as a *multi layer network* [26].

It should also be noted that applications of community detection algorithms are not limited to ECU network architecture design. For example, a related problem is the distribution of functions into ECUs. ECUs need to communicate because functions in one ECU depend on data provided by functions contained in other ECUs. There is thus a network of connectivity between functions, similar to that between ECUs, but with many more nodes. Running community detection algorithms on this network would suggest ways of grouping functions together into ECUs. At the moment, we understand JLR does not have this freedom to move functions between ECUs, but in future designs they may have.

ACKNOWLEDGEMENTS

We would like to thank Samuel Johnson and Harita Joshi for their useful help and input throughout the project. We would also like to thank Gunwant Dhadyalla, Alexandros Mouzakitis, George Pappas, Charo Del Genio and the EPSRC.

REFERENCES

[1] J. Simpson and M. Simpson, "System of systems complexity identification and control," in *Proc. IEEE International Conference on System of Systems Engineering*, Albuquerque, NM, May 2009.

[2] S. M. Mahmud. In-vehicle network architecture for the next-generation vehicles. [Online]. Available: http://ece.eng.wayne.edu/~smahmud/PersonalData/PubPapers/IGI-Book-2009.pdf

[3] M. Newman, *Networks:An Introduction.* Oxford: Oxford University Press, 2015, p. 358.

[6]On a practical level, it is understood that message ID's often change when they pass through a gateway - a complete dataset would need to trace these changes.

[4] "Easis general architecture framework. d0.2.4." Aug 2004.

[5] U. Alon, "Public dataset," http://wws.weizmann.ac.il/mcb/UriAlon/index.php?q=download/collection-complex-networks, 2004, [Online; accessed 7-May-2015].

[6] S. Fortunato, "Community detection in graphs," *Physics Reports*, p. , 2010.

[7] Wikipedia, "Small network," 2011, [Online; accessed 11-June-2015]. [Online]. Available: http://en.wikipedia.org/wiki/Community_structure#/media/File:Network_Community_Structure.svg

[8] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, "Defining and identifying communities in networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 101, no. 9, pp. 2658–2663, 2004. [Online]. Available: http://www.pnas.org/content/101/9/2658.abstract

[9] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Phys. Rev. E*, 2004.

[10] E. A. Leicht and M. E. J. Newman, "Community Structure in Directed Networks," *Physical Review Letters*, vol. 100, no. 11, p. 118703, Mar. 2008.

[11] M. E. J. Newman, "Analysis of weighted networks," *Phys. Rev. E*, vol. 70, p. 056131, Nov 2004. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevE.70.056131

[12] S. Fortunato and M. Barthlemy, "Resolution limit in community detection," *Proceedings of the National Academy of Sciences*, vol. 104, no. 1, pp. 36–41, 2007. [Online]. Available: http://www.pnas.org/content/104/1/36.abstract

[13] M. Rosvall and C. Bergstrom, "An information-theoretic framework for resolving community structure in complex networks," *Proc. Natl. Acad. Sci. USA*, vol. 104, no. 18, pp. 7327–7331, 2007.

[14] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, p. , 2008.

[15] S. Sobolevsky, R. Campari, A. Belyi, and C. Ratti, "A general optimization technique for high quality community detection in complex networks," *CoRR*, vol. abs/1308.3508, 2013. [Online]. Available: http://arxiv.org/abs/1308.3508

[16] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell System Technical Journal*, vol. 49, no. 2, pp. 291–307, 1970. [Online]. Available: http://dx.doi.org/10.1002/j.1538-7305.1970.tb01770.x

[17] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Phys. Rev. E*, vol. 78, p. 046110, Oct 2008. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevE.78.046110

[18] A. Lancichinetti and S. Fortunato, "Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities," *Phys. Rev. E*, vol. 80, p. 016118, Jul 2009.

[19] ——, "Community detection algorithms: A comparative analysis," *Phys. Rev. E*, vol. 80, p. 056117, Nov 2009. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevE.80.056117

[20] L. Danon, A. Daz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2005, no. 09, p. P09008, 2005.

[21] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. New York: Cambridge University Press, 2007, p. 549.

[22] E. Cuthill and J. McKee, "Reducing the bandwidth of sparse symmetric matrices," in *ACM '69 Proceedings of the 1969 24th national conference*, pp. 157–172.

[23] M. Tan Khoa Vo, "Exact and heuristic solutions to the bandwidth minimization problem," Master's thesis, Heidelberg University, Oct 2011. [Online]. Available: http://archiv.ub.uni-heidelberg.de/volltextserver/12580/1/KhoaVo_Dissertation.pdf

[24] A. Pothen, H. D. Simon, and K.-P. Liou, "Partitioning sparse matrices with eigenvectors of graphs," *SIAM J. Matrix Anal. Appl.*, vol. 11, no. 3, pp. 430–452, May 1990. [Online]. Available: http://dx.doi.org/10.1137/0611030

[25] P. Kleberger, N. Nowdehi, and T. Olovsson, "Towards designing secure in-vehicle network architectures using community detection algorithms," in *Vehicular Networking Conference (VNC), 2014 IEEE*, Dec 2014, pp. 69–76.

[26] M. Kivelä, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, and M. A. Porter, "Multilayer Networks," *ArXiv e-prints*, Sep. 2013.