Design Optimization of Vehicle Control Networks
Seong-Woo Kim, Eun-Dong Lee, Mid-Eum Choi, Han-You Jeong, and Seung-Woo Seo

Abstract—The advancement of electronic technology has made significant contributions to the safety and convenience of modern vehicles. New intelligent functionalities of vehicles have been implemented in a number of ECUs that are connected to each in vehicle control networks (VCNs). However, with the rapid increase in the number of ECUs, VCNs are currently facing several challenges such as design complexity, space constraint, system reliability, inter-dependency, etc. Considering these factors, the complexity of the VCN design problem increases exponentially, which means the problem cannot be solved within a reasonable time using conventional optimization techniques.

In this paper, we report a new methodology for the optimal design of VCNs. An analytical model was derived to examine the fundamental characteristics of the VCN design problem. Compared to the case of a conventional data network which typically considers temporal scheduling over a fixed physical topology, the VCN design problem should also consider spatial constraints, such as the volume, position and weight. Moreover, the spatial constraints change during the solving procedure. Such temporal and spatial joint optimization problems with varying constraints incur extremely high computational complexity. To tackle the high complexity, this paper proposes a fast solution based on a repeated matching method, which reduces the problem complexity from $O(N^{2N})$ to $O(N^N)$. By applying our methodology to a number of different real-world VCN design scenarios, this proposal can produce a 1% near-optimal design within a significantly reduced time.

Index Terms—Design optimization, in-vehicle communication, repeated matching method, simulated annealing, topological optimization, vehicle control network.

I. INTRODUCTION

As more functionality related to safety and convenience is integrated into a vehicle system, the number of electronic control units (ECUs) is increasing, and can range from a dozen up to 100, supporting many subsystems, such as cruise control, powertrain management, suspension control, etc. [1]. Distributed ECUs are typically connected to each other via vehicle control networks (VCNs). The well-known VCN technologies include LIN [2], CAN [3], Flexray [4] or Byteflight [5]. Among them CAN is the most widely adopted for vehicle control purposes. However, with the rapid increase in the number of ECUs, VCNs are currently facing several challenges.

First, the design complexity originating from the ECU inter-dependency has increased significantly according to the growth of new vehicle functionalities. Second, physical spatial limitations have become a significant bottleneck of VCN design to adopt newly required ECUs. Third, as the safety requirement increases, hardware and software redundancy is needed in VCN design. However, redundancy is normally hindered by space limitations and incurs higher design complexity. Lastly, VCN design is required to be completed earlier than before, due to the shortened new-vehicle development cycle. However, typical VCN design is a very time-consuming process and it normally takes a long time to obtain and verify a result considering the above factors.

Figure 1 shows a general design process of a VCN. In the first step, engineers design functional blocks to satisfy the requirements. This step typically follows the requirements from the use cases or application scenarios within a vehicle. In the second step, for each functionality, a part of a functional block, is decomposed into a number of tasks. In the third step, the tasks are allocated to specific ECUs. Each ECU is located at a specific position of the vehicle subsystems. Finally, ECUs are positioned on particular VCNs. Therefore, the aim of the VCN design problem is to determine the optimal mapping between tasks and ECUs, and between ECUs and VCNs within a bounded time, while meeting all the requirements and constraints.

To achieve this goal, this paper first provides an analytical model to examine the fundamental characteristics of the VCN design problem. However, although the problem is well-modeled analytically, it is still not feasible or extremely difficult to obtain a satisfactory design result due to the high complexity of the problem. In principle, finding an optimal
solution of the above problem is NP-hard, because the optimal solution can be calculated in \(O(J^K N)\) time, where \(N, K\) and \(J\) denote the number of tasks, ECUs, and candidate ECU locations, respectively.

Moreover, the spatial constraints may change during the solving procedure of the VCN design problem. Spatial constraints refer to the requirements: e.g., some ECUs must be positioned at particular locations and additional ECUs cannot be accommodated due to the limited vehicle volume. With these constraints, the results of task allocation can affect significantly the number of ECUs to be adopted. Moreover, the ECU position must be redetermined along with the change in task allocation. Since the change in ECU position affects the ECU network topology directly, it is important to conform whether the temporal requirements, such as controllability and schedulability are still satisfied (see Table I). The above processes are repeated until a good design is obtained, which incurs extremely high computational complexity, and cannot be solved within a reasonable time using conventional optimization techniques.

A new tractable approach was developed to tackle this complex spatial and temporal joint optimization problem. In particular, a fast solution methodology based on a repeated matching technique was originally invented by Wark et al. [6] to solve a vehicle routing problem within a polynomial time. This method can reduce the computational complexity from \(O(J^K N)\) to \(O(J^K)\). This test with different real-world vehicle design scenarios demonstrated that 1% near-optimal results can be obtained within a significantly reduced time.

The main contribution of this study can be summarized as follows:

- **Defining a new problem**: The VCN design problem is decomposed into the following decision subproblems: 1) task allocation problem (TAP), 2) ECU positioning problem (EPP), and 3) network assignment problem (NAP).

- **Presenting a new analytical model**: A detailed analytical model for each subproblem is provided to tackle the above three subproblems. An integrated model considering temporal and spatial factors are presented simultaneously.

- **Providing a fast solution**: Based on these models, a fast method is proposed to determine the optimal design within the computational complexity \(O(J^K)\), which is low complexity because of \(J < K \ll N\), typically.

- **Achieving high accuracy**: The proposed method can obtain a 1% near-optimal solution for real-world scenarios.

The remainder of this paper examines the VCN design in more detail. Section II outlines related work. Section III, IV and V describe TAP, EPP and NAP, respectively. Section VI explains the corresponding resolution approaches in detail. Section VII provides a comparative performance evaluation with conventional approaches. Section VIII concludes this study.

### II. RELATED WORK

Many studies have examined ways of solving the above VCN design problems independently. Table I lists the related studies organized according to their target and methodology. The early-stage works on VCN design optimization focused on one aspect of the following: temporal (e.g., schedulability, availability), spatial (e.g., topology, ECU positioning), reliability (e.g., fault tolerance, redundancy), etc.

A number of recent studies have been proposed to optimize the VCN design considering more than one aspect simultaneously. Łukasiewycz et al. proposed a topology optimization to minimize the number of network devices while satisfying the timing constraints [10]. Glaß et al. proposed an optimal ECU positioning method considering the reliability [14]. Izosimov et al. attempted to maximize the reliability while satisfying the schedulability requirements [18]. Kopetz provided experimental results of sending additional physical messages per one logical message to enhance the VCN reliability [21].

Several methodologies have been also proposed to solve the VCN design problems based on industrial softwares such as AUTOSAR (AUTomotive Open System ARchitecture) [22]. Scheickl et al. well organized and addressed VCN design issues especially considering timing aspect and task mapping in AUTOSAR specification [23]. However, the authors provided no specific methodology to solve the problems. Reichelt et al. introduced design methodology for Flexray-based automotive systems [24]. However, the authors focused on timing aspects only. Current AUTOSAR release does not sufficiently cover the adequate timing aspects of the automotive system [25].

Overall, many studies have examined VCN design optimization. However, no integrated VCN design methodology for temporal and spatial joint optimization with a fast solution method has been published. The remainder of the paper focuses on developing a fast solution for the temporal and spatial joint optimization problem considering the varying spatial constraints and reliability at the same time.

### III. PROBLEM FORMULATION OF TASK ALLOCATION

This section examined the task allocation problem (TAP) as the first step of VCN design. The corresponding analytical model is provided to formulate the TAP.
The total cost is formulated as follows:

\[ T = \sum_{i \in \mathcal{E}} q_{ij} y_{ij} + \sum_{j \in \mathcal{E}} e_j x_j \]  

where \( \mathcal{E} \subseteq \mathcal{E}_A \). The objective function (1) consists of two cost terms. The left term represents the total cost installing tasks into ECUs. \( y_{ij} \) is a binary variable equal to 1 if \((i,j) \in \mathcal{E}_A\); otherwise it is equal to 0. \( q_{ij} \) is a cost incurred by installing \( T_i \) to \( E_j \), and can be formulated as

\[ q_{ij} = \sum_{k \in U} q_{ij}^k \quad i \in \mathcal{T}, j \in \mathcal{E} \]  

where \( U \) is the set of cost factors such as wiring installation cost, network overhead, etc. \( q_{ij}^k \) denotes the \( k \)th cost factor in \( U \) incurred when \( E_j \) accommodates \( T_i \).

For practical purposes, it is normally better to represent \( q_{ij}^k \) using a marginal installation cost \( \Delta q_{ij}^k \). This is motivated by the fact that commercially produced ECUs have their task list in most real-world vehicle systems, which means that initial task-to-ECU mappings are given. Therefore, it is more beneficial to deal with marginal costs instead of an absolute value \( q_{ij}^k \). \( \Delta q_{ij}^k = q_{ij}^k - q_{ij}^0 \) if a task \( T_i \) is initially allocated to an \( E_j \). Similarly, \( \Delta q_{ij} = q_{ij} - q_{ij}^0 \). \( \Delta q_{ij} \) can be interpreted as a marginal cost to install \( T_i \) into \( E_j \) compared to the initially allocated ECU. In the case mapping is not given, a greedy-mapping can be used as the initial condition.

For example, if \( k \) represents a wiring cost, \( \Delta q_{ij}^k \) means the varying wiring cost incurred by moving \( T_i \) from initially allocated ECU to \( E_j \) in perspective of cost factor \( k \), \( k \in U \); \( D_i \) number of \( T_i \)’s replication; \( x_j \) binary variable equal to 1 if at least one task resides in \( E_j \), otherwise 0; \( y_{ij} \) binary variable equal to 1 if \( T_i \) is located at \( E_j \), otherwise 0.

Note that \( x_j \) and \( y_{ij} \) are decision variables whereas others are given. In addition, let \( | \cdot | \) denote the cardinality of the inner set, e.g., \( N = |T| \) and \( K = |E| \).

**B. Objective Function**

Let \( \mathcal{G}(V, E) \) be the directed graph with vertex set \( V \) and edges \( E \). \( \mathcal{V} = T \cup E \) denotes the union set of tasks and ECUs. In \( \mathcal{G} \), a directed edge from \( T_i \) to \( E_j \) denoted as \((T_i, E_j)\) shows that \( T_i \) is allocated to \( E_j \). Let \( \mathcal{E}_A = \{(i,j) \mid i \in \mathcal{T}, j \in \mathcal{E}\} \) be the set of all possible edges.

TAP eventually aims to determine the optimal subset of \( \mathcal{E}_A \). The total cost is formulated as follows:

\[ f(\mathcal{E}) = \sum_{i \in \mathcal{T}} \sum_{j \in \mathcal{E}} q_{ij} y_{ij} + \sum_{j \in \mathcal{E}} e_j x_j \]

where \( \mathcal{E}(\subseteq \mathcal{E}_A) \). The objective function (1) consists of two cost terms. The left term represents the total cost installing tasks into ECUs. \( y_{ij} \) is a binary variable equal to 1 if \((i,j) \in \mathcal{E}_A\); otherwise it is equal to 0. \( q_{ij} \) is a cost incurred by installing \( T_i \) to \( E_j \), and can be formulated as

\[ q_{ij} = \sum_{k \in U} q_{ij}^k \quad i \in \mathcal{T}, j \in \mathcal{E} \]  

where \( U \) is the set of cost factors such as wiring installation cost, network overhead, etc. \( q_{ij}^k \) denotes the \( k \)th cost factor in \( U \) incurred when \( E_j \) accommodates \( T_i \).

For practical purposes, it is normally better to represent \( q_{ij}^k \) using a marginal installation cost \( \Delta q_{ij}^k \). This is motivated by the fact that commercially produced ECUs have their task list in most real-world vehicle systems, which means that initial task-to-ECU mappings are given. Therefore, it is more beneficial to deal with marginal costs instead of an absolute value \( q_{ij}^k \). \( \Delta q_{ij}^k = q_{ij}^k - q_{ij}^0 \) if a task \( T_i \) is initially allocated to an \( E_j \). Similarly, \( \Delta q_{ij} = q_{ij} - q_{ij}^0 \). \( \Delta q_{ij} \) can be interpreted as a marginal cost to install \( T_i \) into \( E_j \) compared to the initially allocated ECU. In the case mapping is not given, a greedy-mapping can be used as the initial condition.

For example, if \( k \) represents a wiring cost, \( \Delta q_{ij}^k \) means the varying wiring cost incurred by moving \( T_i \) from initially allocated ECU to \( E_j \) in perspective of cost factor \( k \), \( k \in U \); \( D_i \) number of \( T_i \)’s replication; \( x_j \) binary variable equal to 1 if at least one task resides in \( E_j \), otherwise 0; \( y_{ij} \) binary variable equal to 1 if \( T_i \) is located at \( E_j \), otherwise 0.

Note that \( x_j \) and \( y_{ij} \) are decision variables whereas others are given. In addition, let \( | \cdot | \) denote the cardinality of the inner set, e.g., \( N = |T| \) and \( K = |E| \).

**C. Constraints**

1) Resource Constraint: This indicates that an ECU must be able to support the resource request of the residing tasks.

\[ \sum_{i \in \mathcal{T}} s_i y_{ij} \leq c_j \quad \forall j \in \mathcal{E} \]  

where \( Y_{ij} \) is the identity matrix if \( y_{ij} = 1 \); otherwise it is a zero matrix. Let \( A \) denote the set of computing
resources, such as a processor for computation or memory for storage. If $a_1$ represents a memory, $s_i^{a_1}$ denotes the amount of memory required by $T_i$. $s_i$ denotes the resource demand vector of $T_i$, i.e., $s_i = [s_i^{a_1}, \ldots, s_i^{a_{|A|}}], a_k \in A$. Similarly, $c_j^{a_1}$ denotes the total amount of memory supported by $E_{ij}$, $c_j$ denotes the resource capacity vector that $E_{ij}$ can support, i.e., $c_j = [c_j^{a_1}, \ldots, c_j^{a_{|A|}}], a_k \in A$. The piecewise inequality (3) is intended for a generalized matrix model to support multiple resources.

If a single resource is considered, (3) can be written as $\sum_{i \in T} s_i^{a_k} y_{ij} \leq c_j^{a_k}$.

2) Assignment Constraint: This constraint indicates that a task needs to be located in at least one ECU.

$$\sum_{j \in E} y_{ij} = D_i \quad \forall i \in T \tag{4}$$

where $D_i$ is the number of $T_i$’s redundant allocations. $D_i$ is related directly to the reliability of $T_i$ and a system including the task. $D_i = 1$ if $T_i$ has no redundant task allocation. $D_i = 2$ if one task for $T_i$ has to be replicated once. To guarantee the system availability from the Byzantine faults, $D_i$ has to be larger than three [28].

3) ECU Open Constraint: This indicates that $x_j$ is a binary variable equal to 1 if $E_j$ has more than one task; otherwise it is equal to 0.

$$y_{ij} \leq x_j \quad \forall i \in E, \quad \forall j \in T \tag{5}$$

4) Binary Constraint: This indicates that the eventual solution of this section is to know which task should be allocated to which ECU.

$$y_{ij} \in \{0, 1\} \quad x_j \in \{0, 1\} \quad \forall i \in E, \quad \forall j \in T. \tag{6}$$

D. Problem Formulation

TAP can be represented as follows:

$$\text{minimize} \quad (1)$$

subject to \quad (2), (3), (4), (5) and (6). \tag{7}

As a result of solving (7), optimal task-to-ECU mapping $E^*$ can be obtained, which is defined as

$$f(E^*) \leq f(E) \quad \forall E.$$

**Lemma 1**: $f(E) = f_\Delta(E) + C$, where $C$ is constant.

**Proof**: Let $\hat{q}_i$ denote an installation cost of $T_i$ to an initially allocated ECU. Because of $\sum_{j \in E} y_{ij} = D_i$ by (4),

$$\sum_{i \in T} \sum_{j \in E} \Delta q_{ij} y_{ij} = \sum_{i \in T} \sum_{j \in E} \Delta q_{ij} y_{ij} = \sum_{i \in T} \sum_{j \in E} (q_{ij} - \hat{q}_i) y_{ij} = \sum_{i \in T} \sum_{j \in E} q_{ij} y_{ij} - \sum_{i \in T} \hat{q}_i \sum_{j \in E} y_{ij} = \sum_{i \in T} \sum_{j \in E} q_{ij} y_{ij} - \sum_{i \in T} \hat{q}_i D_i.$$

Let $\sum_{i \in T} \hat{q}_i D_i$ be a constant $C$ when $\hat{q}_i$ and $D_i$ are given.

In that case, $f_\Delta(E) = f(E) - C$.

By Lemma 1, $f(E)$ can be reformulated as

$$f(E) = f_\Delta(E) + C. \tag{8}$$

Using (8), we can reform (7) to

$$\text{minimize} \quad f_\Delta(E)$$

subject to \quad (2), (3), (4), (5) and (6). \tag{9}

**Lemma 2**: (7) is equivalent to (9).

**Proof**: The objective function of (7), i.e., $\min f(E)$, can be reformed to $E^* = \arg\min f(E) = \arg\min f_\Delta(E) + \arg\min C = \arg\min f_\Delta(E) + C$. Constant value $C$ has no influence on the $y_{ij}$ decision.

By Lemma 2, (9) can be used interchangeably for an optimal solution of (7).

**Proposition 1**: The task allocation problem of vehicle control networks is NP-hard.

**Proof**: Task allocation to ECU corresponds to customer allocation to the facility in the facility location problem (FLP). The installation cost of facility and customer’s transportation cost of the FLP correspond to the installation cost $c_j$ and $q_{ij}$ respectively. The TAP of a VCN is an example of the facility location problem (FLP). FLP is known as NP-hard to solve optimally. The proof is completed.

According to Proposition 1, the globally optimal solution of the task allocation problem can be calculated in $O(K^N)$ time, where $K$ and $N$ denote $|E|$ and $|T|$, respectively. The problem-solving approach to solving (7) and (9) within polynomial time is provided in Section VI.

IV. ECU POSITIONING PROBLEM

Since the result of solving TAP affects the number of ECUs, the ECU may be re-located on a vehicle after obtaining a certain result from TAP. This section formulates the ECU positioning problem (EPP) with the constraint of wiring consisting of sub-lines and main-lines.

A. Assumption and Overview

The approaches introduced in this section are not affected significantly by any specific VCN technology. To intuitively specify the proposed concepts in this and next section, it was assumed that the target vehicle system uses CAN. This assumption is normally feasible and practical, because CAN is currently a de facto standard of VCN technology due to license-free usage, cost-efficiency and verified performance.

Fig. 3 shows the concept of the ECU network topology consisting of a stub-line and main-line. The length of the stub-line is limited to a particular length $S$, which is specified depending on its operating speed in the standard [29], [30].
For transmission speeds up to 1 Mbit/s in CAN, the maximum allowable stub-line length $S$ is 0.3 m. $S$ can be increased theoretically to 6 m according to decrease in bus speed.

If the length of a line for connecting ECU to the main-line is larger than $S$, the main-line needs to be extended in the ECU direction by as much as $d_j - S$ in Fig. 3. $d_j$ denotes the length of wiring connecting a location $j$ to the main-line. However, since the main-line typically resides in a common cable path, extending the common cable path incurs considerable cost. Thus, the goal of EPP is to determine the optimal position for given ECUs while minimizing the wiring installation cost.

B. General Description and Notation

The following defines the notations for the ECU positioning problem:

- $O$: set of locations where an ECU can be located;
- $O_i$: set of locations where $E_i$ can be located;
- $h_{ij}$: installation cost when an ECU is located at location $j$;
- $v_i$: volume of $E_i$;
- $\psi_j$: maximum volume capacity of location $j$;
- $G_{\text{init}}$: length of an initial main-line;
- $G_{\text{max}}$: maximum length of a main-line;
- $S$: maximum allowable length of a stub-line;
- $v_{\text{bus}}$: speed of electron;
- $b_{\text{bus}}$: baud rate of bus;
- $\beta$: reference electrical wire cost for a stub-line per meter;
- $\gamma$: cost due to extending a main-line per meter;
- $d_j$: length of a wiring for connecting between a location $j$ and the main-line;
- $l_j$: length of a stub-line of an ECU located in $j \in O$;
- $m_j$: length of an extended main-line to connect the stub-line of an ECU located in $j \in O$;
- $\sigma_j$: scale factor representing how much a main-line should be extended for a ECU located at a location $j$;
- $k_{ij}$: binary variable equal to 1 if $E_i$ is located at location $j$, otherwise it is equal to 0.

Note that $k_{ij}$ is a decision variable, while the others are given.

C. Objective Function

The cost function of an EPP can be formulated as follows:

$$
\min \sum_{i \in E} \sum_{j \in O_i} h_{ij} k_{ij}
$$

where $h_{ij}$ is a wire installation cost that an ECU is located at location $j$. $h_{ij}$ can be represented as

$$
h_{ij} = \beta l_j + \gamma m_j
$$

where

$$
l_j = \begin{cases} 
    d_j & d_j \leq S \\
    S & \text{otherwise}
\end{cases}
$$

D. Constrains

1) Volume Constraint: The space capacity of a location $j$ is given $\phi_j$. Assume $E_i$ occupies $v_i$ amount of space.

$$
\sum_{i \in E} v_i k_{ij} \leq \phi_j \quad \forall j \in O.
$$

2) Main-line Constraint: The maximum length of a main-line should not exceed $U_{\text{bus}}$.

$$
G_{\text{init}} + \sum_{i \in E} \sum_{j \in O} \sigma_j m_j k_{ij} \leq G_{\text{max}}.
$$

3) Timing Constraint: The topology of the stub lines and main-line affects the timing aspects. Assuming that the control network technology is CAN, the timing constraint is derived as follows:

$$
l_v + l_w + G_{\text{init}} + \sum_{i \in E} \sum_{j \in O} \sigma_j m_j k_{ij} \geq v_{\text{bus}} \left( \frac{\delta}{2b_{\text{bus}}} - t_d \right)
$$

where $v, w \in O \subseteq O$. $O$ is the set of locations hosting at least one ECU in a candidate $k_{ij}$ set. $\delta$ denotes a fraction of sample point in bit time. $t_d$ is the total transmission delay of a CAN signal. The detailed derivation can be seen in the Appendix C. This constraint means the wiring can be modified until schedulability is guaranteed. However, even if the adopted VCN can be changed to other technologies, it is sufficient to slightly modify only this timing constraint according to physical characteristics of the communication technology.

4) ECU Assignment Constraint: This indicates that a task has to be located in a single location.

$$
\sum_{j \in O_i} k_{ij} = 1 \quad \forall i \in E.
$$
5) Binary Constraint:

\[ k_{ij} \in \{0, 1\} \quad \forall i \in E, \quad \forall j \in O. \tag{16} \]

In practice, the number of ECUs and candidate ECU locations are typically less than a hundred. In this case, conventional assignment methods are sufficient to obtain the solution of (10), (11), (12), (13), (14), (15), and (16).

V. NETWORK ASSIGNMENT PROBLEM

Since the result of TAP and EPP affects the network performance, the feasibility must be validated in term of the network schedulability. This step is called the network assignment problem (NAP). The aim of NAP is to confirm that the results of TAP and EPP satisfy the network requirements. As explained in Sec. III.A, CAN is assumed in this paper for the sake of an explanation of NAP. This section formulates the NAP based on CAN. However, the fundamental methodology is not changed, even if the technology is altered to others, such as Flexray [31], [32] or TTCAN [33].

A. General Description and Notation

The following defines the notations of the NAP in advance:

- \( B \) set of buses;
- \( M \) set of messages, where \(|M| = I\);
- \( B_m \) bus on which a message \( m \) is passed;
- \( E^T_m \) set of target ECUs of message \( m \);
- \( p_m \) period of message \( m \);
- \( w_m \) priority of message \( m \);
- \( g_m \) generation delay of message \( m \);
- \( t_m \) queuing delay of message \( m \);
- \( u_m \) transmission delay of message \( m \);
- \( F_m \) blocking time incurred by messages with lower priority than the message \( m \);
- \( r_m \) worst case response time of message \( m \);
- \( f_m \) deadline of message \( m \).

B. System Model

Consider a bus \( b \in B \) shared by \( I \) messages. Only one message can be transmitted at a time through the bus. Suppose that particular buses satisfy \( B_{m_1} \cup B_{m_2} = \emptyset, m_1, m_2 \in M \). If there is a gateway relaying messages between \( B_{m_1} \) and \( B_{m_2} \), \( B_{m_1} = B_{m_2} \). Each message has two parameters; period \( p_m \) and priority \( w_m \). For an intuitive understanding, it is assumed that

\[ w_i < w_j \quad \text{when} \quad i < j \]

where \( i, j \in M \), which means higher priority message has a higher message number. There are several methods to give a priority \( w_m \) to a message \( m \). For example, roughly \( w_i = I - i \) in CAN [3]. \( w_i = i \) is also used widely. Consequently, a message \( m \) has predetermined priority \( w_m \) and period \( p_m \).

The priority is used for bus sharing in distributed networks, particularly in distributed control networks. A task attempts to transmit messages according to their predetermined periods. Suppose that a number of messages attempt to be transmitted at a certain time \( t \). The messages are denoted by \( M \subset M \).

If more than two messages attempt to access the bus at the same time, only the highest priority message \( m_h \in M \) is allowed to access the bus. The remainder of the messages, i.e., \( \{M - m_h\} \), need to wait until \( m_h \) finishes being transmitted. In other words, the transmitted message is not preemptive. Note that typically \( w_i > w_j \), if \( p_i < p_j \), even though the period and priority can be given independently to a specific message [34].

C. Schedulability Condition

There are two conditions to satisfy the schedulability requirements of in-vehicle communications.

1) Condition 1: In a specific control network, a message must be delivered within a predetermined deadline even in the worst case. The constraint is represented as follows:

\[ r_m \leq f_m, \quad \forall m \in M \]

where \( f_m \) is the predetermined deadline of message \( m \). The worst case response time \( r_m \) of a message \( m \) consists of three components: generation \((g_m)\), queuing \((t_m)\) and transmission delays \((u_m)\). In other words, \( r_m = g_m + t_m + u_m \). The calculation method for \( r_m \) is explained in the next subsection.

2) Condition 2: The source and destination ECU of a message needs to be assigned to the same bus. The following must be satisfied:

\[ E^T_m \subseteq B_m, \quad \forall m \in M. \]

If the two conditions are not satisfied, TAP and EPP need to be conducted again excluding the current parameters.

D. Schedulability Assessment

The scheduling model of CAN is well-organized by Tindell et al. [35]. Although some slightly improved models have been proposed such as [36], the model reported by Tindell still works well. Hence, the Tindell model is used for the CAN schedulability assessment. The main part of CAN schedulability can be formulated as

\[ g_m = u_m + F_m + \sum_{l \in hp(m)} \left\lfloor \frac{g_m - u_m}{p_l} \right\rfloor u_l \]

where \( p_l \) denotes the period of message \( l \), and \( hp(m) \) denotes the set of higher priority messages than message \( m \). \( u_m \) and \( u_l \) represent the execution time of message \( m \) and \( l \) in the worst case [37], [38]. \( \left\lfloor \frac{g_m - u_m}{p_l} \right\rfloor \) denotes the maximum number of interferences from the higher priority messages of task \( l \). \(-u_m\) means that a transmitting message is not preemptive. \( F_m \) is the maximum time waiting for the end of currently transmitting messages with lower priority than message \( m \).

\[ F_m = \max_{l \in p(m)} u_l, \quad \text{where} \quad p(m) \text{ denotes the set of lower priority messages than message } m. \]

VI. RESOLUTION APPROACH

Based on the analytical models in previous sections, the following provides the solving approach in this section.
A. Overview

As mentioned, although VCN design problems are well-formulated analytically, it is still difficult to obtain the optimal solution within a reasonable time due to the huge amount of searching space, i.e., $O(J^NK^3)$. In particular, the bottleneck is a process of solving TAP, because TAP requires $O(KN)$ computational complexity and $J < K << N$. TAP can be solved using the conventional integer linear programming (ILP) solver, but the formulation in (7) is a form of ILP. However, the ILP solver takes a substantial time to calculate the optimal solution.

Many studies have been performed to obtain the optimal allocation rapidly, such as TAP in operational research. Among them, Rönnqvist et al. examined the optimal facility for each customer and optimal airplane for each air-crew in [39], [40]. To determine the optimal allocation, they proposed an approach to convert the form of (7) to a matching problem (MP). The approach is called the repeated matching method. Using MP, the computational complexity is reduced from $O(J^NK^3)$ to $O(K^3)$. Moreover, MP can be solved within approximately $O(K^3)$ using the successors of the Hungarian method [41] that is based on the fact that the assignment problem (AP) can solved using various approaches based on the Hungarian method with $O(K^3)$ computation complexity. MP is a relaxation of AP with symmetric constraints. As a result, the computational complexity can be reduced to $O(J^NK^3)$. Figure 4 shows the computational complexity reduction of this proposal when the repeated matching method is applied. The method repeats until an optimal solution is obtained by generating a better MP formulation in each step based on the one from the previous step. The following shows how to reformulate the VCN optimization problem, as an MP and provides a fast solution based on the repeated matching method.

B. Matching Problems

The repeated matching method used to solve the problem (7) is based on an MP defined as follows:

$$\begin{align*}
\min & \sum_{i} \sum_{j} b_{ij} z_{ij} \\
\text{s.t.} & z_{ij} = 1, i = 1, ..., k, \\
& \sum_{i=1}^{k} z_{ij} = 1, i = 1, ..., k, \\
& z_{ij} = z_{ji}, i = 1, ..., k, \\
& z_{ij} \in \{0, 1\}, i, j = 1, ..., k,
\end{align*}$$

(17)

where $b_{ij}$ denotes the cost for matching $i$ with $j$, and $b_{ij} = b_{ji}$. $z_{ij}$ is 1 if $i$ is matched with $j$; otherwise it is equal to 0. $z_{ij}$ is a symmetric constraint and $z_{ij} \in \{0, 1\}$ is a binary constraint. Without the symmetric constraint $z_{ij} = z_{ji}$, (17) is an assignment problem, which can be formulated as an ILP, and can be solved using ILP solvers. To solve (7) using this MP method, the matching cost $b_{ij}$ needs to be constructed from (7).

C. Matching Costs

Define matching cost matrix $b$ as follows:

$$b = \begin{bmatrix}
L_1 - L_1 & [-] & [-] \\
L_2 - L_1 & [L_2 - L_2] & [-] \\
L_3 - L_1 & [L_2 - L_2] & [L_3 - L_3]
\end{bmatrix}$$

where $L_1$ is the set of all ECUs that are unused, and $L_2$ is the set of tasks that are not assigned to an ECU. The set $L_3$ consists of all ECUs used with their assigned tasks. For example, $L_1 = \{E_2\}, L_2 = \{T_3\}$, and $L_3 = \{(E_1, \{T_1, T_2\}), (E_3, \{T_6, T_7\}), (E_4, T_3)\}$, in Fig. 2.

We define an assignment $(j, R)$ as feasible if $j \in E$ and $\sum_{i \in R} s_i \leq c_j$, $R \subset T$. In other words, an assignment is feasible if an ECU can accommodate the total demand of tasks hosted by the ECU. Therefore, all elements of $L_3$ must be feasible and

$$(j_1, R_1), (j_2, R_2) \in L_3 \quad \text{if} \quad R_1 \cap R_2 = \emptyset \quad \text{and} \quad j_1 \neq j_2.$$
means that \( i \) is matched with \( j \). As the iteration proceeds, the cost of (17) is reduced. This optimization process is repeated until no improvement is made, namely \( 1^T b(x+1)z(x+1) = 1^T b(x)z(x) \).

The detailed method to calculate the matching cost \( b \) is provided in Appendix A.

**D. Repeated Matching Method**

**Algorithm 1** Repeated Matching Algorithm

1. \( L_1^{(0)} \leftarrow E, L_2^{(0)} \leftarrow T, L_3^{(0)} \leftarrow \emptyset \)
2. \( \text{gopt} \leftarrow \infty, \tau \leftarrow T_0 \)
3. while \( \tau > \tau_{th} \) do
4. compute matching matrix \( b^x \) with \( L_1^{(x)}, L_2^{(x)}, L_3^{(x)} \)
5. \( z^{(x)} \leftarrow \arg \min_z 1^T b^{(x)} z \)
6. \( \text{lopt} \leftarrow 1^T b^{(x)} z \)
7. if \( \text{gopt} > \text{lopt} \) then
8. \( \text{gopt} \leftarrow \text{lopt} \)
9. \( \left( L_1^{(x)}, L_2^{(x+1)}, L_3^{(x+1)} \right) = \arg \min \left( z^{(x)} \right) \)
10. \( x \leftarrow x + 1 \)
11. else
12. \( b \leftarrow \text{false} \)
13. for \( i \in L_3 \) do
14. if \( e^{-\beta(\text{lopt} - \text{gopt})/\tau} > RND(0, 1) \) then
15. for each \( L_3 \) split into \( L_1, L_2 \)
16. \( b \leftarrow \text{true} \)
17. end if
18. end for
19. if \( b \) then
20. stop algorithm
21. end if
22. end if
23. \( \tau \leftarrow \tau \tau \)
24. end while

A specific solving procedure is given in Algorithm 1. Initially, the algorithm starts by setting all the ECUs as unused, i.e., \( n_1 = |E| \), and all tasks as unassigned, i.e., \( n_2 = |T| \). Next, the matching matrix \( b \) is calculated with the current \( L_1, L_2 \) and \( L_3 \). When the calculation of \( b \) is finished, \( b_{13} \) represents the minimum cost if row \( i \) matches column \( j \). With the matching cost \( b \), the minimum matching problem (17) is calculated next in line 4-5.

Many studies have examined a solving method of MP. However, most are not easy to put into practice because they are too theoretical and complex to implement. In [42], Forbes et al. introduced a practical resolution approach to solve MP based on the algorithm of Engquist [43]. The approach of Forbes begins from the observation that the formulation of a minimum MP is identical to the case of AP with symmetric constraints. A commercial ILP solver is used to solve the MP with the calculated \( b \) where the MP solver is embedded. The main contribution of this paper does not lie in a fast solving MP itself.

When \( z \) is obtained at each step, the corresponding \( L_1, L_2 \) and \( L_3 \) are derived using \( z_\Delta(z) \) as in line 9. Since the decision variables \( y_{ij} \) and \( x_j \) can be determined by \( L_3 \), the total cost \( f(\mathcal{E}) \) can be obtained, which is then compared with the previous cost. When comparing the two costs, the already-calculated \( b \) by Theorem 1 is used, instead of calculating Eq. (1) directly. Theorem 1 suggests that the sum of all diagonal elements of \( b \) is less than or equal to the double of Eq. (1).

**Theorem 1**: Given a set of \( \epsilon_j, q_{ij} \) and \( y_{ij} \),

\[
\sum_{k=1}^{n_1+n_2+n_3} b_{kk} \leq 2 \left( \sum_{i \in T} \sum_{j \in E} q_{ij} y_{ij} + \sum_{j \in E} \epsilon_j x_j \right).
\]  

**Proof**: see Appendix B.

If the calculated cost is reduced than before, new matching has been generated in line 9. When there is no cost improvement, current \( L_1^{(x)} \) and \( L_3^{(x)} \) are the final results.

**E. Simulated Annealing**

When no improvement is made at a certain step of the repeated matching algorithm, a feasible solution may have been obtained. However, it is not guaranteed that the obtained solution is globally optimal. A heuristic method inspired by Simulated Annealing (SA) [44] was used to solve this local minimum problem.

The method first picks a random neighbor \( S' \). If the cost of neighbor \( f_{cost}(S') \) is less than the current minima \( f_{cost}(S) \), the current minimal point \( S \) is updated to the neighbor \( S' \). Otherwise, \( S \) is updated to \( S' \) with the probability \( e^{-\Delta/T} \), where \( \Delta \) denotes the gap between the newly obtained minimum cost and the previous minima, i.e., \( f_{cost}(S') - f_{cost}(S) \). \( T \) denotes the absolute temperature that decreases as the searching process proceeds.

In this method, the physical interpretation of the temperature is inversely proportional to the number of searching rounds. Note that a round is composed of multiple steps \( (x) \), which leads to a local optimum. If the searching process reaches a local minimum, some of \( L_3 \) are chosen with a split probability \( e^{-\beta \Delta/T} \) in line 14. Here, \( \beta \) is the scale coefficient and \( \tau \) corresponds to the temperature. \( \tau \) decreases due to \( r < 1 \) in line 23.

For a selected \( L_3 \), it is split into \( L_1 \) and \( L_2 \). Splitting can generate new matching even though the split process incurs higher cost than the previous local minima. With the new matching set, \( L_1, L_2 \) and \( L_3 \), the above algorithm is applied again until no improvement is made. However, the single split process cannot guarantee that a newly obtained local minimum point is globally optimal. Therefore, this split process may need to be iterated until no further split of \( L_3 \) is possible. The split probability decreases with increasing searching round.

**F. Initial Condition**

The original repeated matching algorithm is initialized with a \( (K+N) \times (K+N) \) matrix. In this case, the worst computational complexity is \( O((K+N)^{K+N}) \). If the set of tasks is genuinely allocated to certain ECUs initially, the repeated matching method begins with \( K \times K \) where \( K \leq K \leq K+N \). For example, the computational complexity can be reduced to \( O(K^N) \) if all tasks are initially allocated to specific ECUs, i.e., \( N = 0 \). Therefore, to make \( N \) zero, this study proposes...
an intelligent method for determining the initial conditions as shown in Algorithm 2.

Algorithm 2 Greedy Method for Initialization
1: \( L_1 \leftarrow E, L_2 \leftarrow T, L_3 \leftarrow \emptyset, T_j \leftarrow \emptyset, \forall j \in E \)
2: for \( i \in L_2 \) do
3: \( \chi \leftarrow E \)
4: \( \nu \leftarrow \arg \min_{j \in E} q_{ij} \)
5: if \( \sum_{k \in \{T_j \cup i\}} s_k \geq c_{ij} \) then
6: \( T_{\nu} \leftarrow T_{\nu} \cup i, L_1 \leftarrow L_1 - \nu, L_2 \leftarrow L_2 - i \)
7: else
8: \( \chi \leftarrow \chi - \nu \)
9: if \( \chi \neq \emptyset \) then
10: goto 4
11: end if
12: end if
13: end for
14: \( L_3 \leftarrow L_3 \cup \{j, T_j\}, \forall j \in \{E - L_2\} \)

In Algorithm 2, \( \nu \) is an ECU with the lowest cost to host a task \( i \). If the ECU has sufficient resources, the task is added to a hosting list of the ECU. These steps are repeated for all tasks from lines 4 to 13. The impact of this initialization is evaluated in the next section.

VII. PERFORMANCE EVALUATION

This section evaluates the proposed methodology with several test scenarios based on different real-world applications in quantitative and qualitative manners.

A. Configuration

Three VCN design scenarios were made to evaluate the feasibility and performance of the proposed methodology. The first scenario is based on the Society of Automotive Engineers (SAE) benchmark for class C application requirement [45] reported by Tindell et al. [46] and Kopetz [21]. The SAE class C application deals with safety-critical applications. The first scenario consists of 54 tasks, 7 ECUs and one high-speed CAN bus [29]. High-speed CAN is used most widely for class C application in practice. The corresponding resource requirements have been determined based on [47]. Every ECU is assumed to be installed at distinct locations within a vehicle. Each task generates a corresponding message except for a task that resides in the instrument panel display ECU and receives other messages without generating any messages.

The second scenario was developed to simulate a modern luxury vehicle that consists of 20 ECUs for safety critical functionalities such as engine management system (EMS), electronic stability program (ESP), steering wheel angle sensor (SAS), transmission control unit (TCU), 4-wheel drive control unit (4WD), dashboard (cluster) unit (CLU), electronic control suspension system (ECS), etc. Among these 20 ECUs, only two receive messages. Therefore, the number of candidate ECUs that must be optimized is 18. All ECUs are connected via a single high-speed CAN where 413 signals are transported over 43 messages in a form of bits.

The third scenario is a mid-size scenario to perform extensive sensitivity analysis and examine the fundamental characteristics of the VCN design problem itself. Scenario 3 is introduced in the next subsection.

Table III summarizes the configurations of scenario 1, 2 and 3.

![Table III: Parameters of Scenarios 1, 2 and 3](http://www.ibm.com/software/integration/optimization/cplex-optimizer/)

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Task</th>
<th>Message</th>
<th>ECU</th>
<th>Location</th>
<th>Bus</th>
<th>Total resource demand (MB)</th>
<th>Total resource capacity (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario 1</td>
<td>54</td>
<td>53</td>
<td>7</td>
<td>4</td>
<td>1</td>
<td>74.6</td>
<td>95</td>
</tr>
<tr>
<td>Scenario 2</td>
<td>413</td>
<td>46</td>
<td>20</td>
<td>13</td>
<td>1</td>
<td>1.971 \times 10^3</td>
<td>2.151 \times 10^3</td>
</tr>
<tr>
<td>Scenario 3</td>
<td>86</td>
<td>-</td>
<td>16</td>
<td>-</td>
<td>1</td>
<td>124</td>
<td>-</td>
</tr>
</tbody>
</table>

B. Parameters of Scenarios 1, 2 and 3

1. \( \beta \): scale coefficient for SA process
2. \( r \): temperature diminishing coefficient
3. \( \tau_0 \): initial temperature
4. \( W_\infty \): penalty cost due to task non-assignment
5. \( \infty \): impossibility

In addition, the maximum cable length was 30 m and the maximum number of nodes was 30 according to ISO 11898.

Since this method does not search all possible solution spaces, the results of this proposal should be compared with the globally optimal solution. A commercial ILP solver, CPLEX\(^1\) was used to obtain the globally optimal solution. All experiments were carried out on an Intel Core 2 Duo Processor 2.66 GHz machine with 2 GB RAM. The solver was implemented using the Java 2 standard edition.

\(^{1}\)http://www.ibm.com/software/integration/optimization/cplex-optimizer/
Sec. III.B showed that it is easier to obtain marginal installation costs. Therefore, the tests were conducted with \( \Delta q_{ij} \) not \( q_{ij} \). \( \Delta q_{ij} \) is assumed to be roughly proportional to the required electrical line length. Table IV lists the common parameter setting of the proposed method. In calculating the matching cost matrix \( b \), it was assumed that the cost is either \( W_\infty \) or \( \infty \) if the tasks are not assigned or matching is infeasible, respectively (see Appendix A in detail).

### B. Performance Comparison

Table V shows the results of these experiments. In scenario 1, the gap between two solutions, one from the ILP solver and the other from the heuristic algorithm, is 1.23%. The computation times for both cases are less than 1 second. In scenario 2, CPLEX obtained a globally optimal solution after approximately three days and seven hours. In contrast, the solver obtained a suboptimal solution with a gap of 0.91% in 46 minutes. All solutions satisfied the schedulability conditions 1 and 2.

Since scenarios 1 and 2 are based on real-world configurations, the results and implication are quite understandable. However, it is difficult to analyze the impact of the parameters and characteristics of the proposed algorithms, because the searching space of scenario 1 is too small, in that it is difficult to observe notable implications. In contrast, scenario 2 takes approximately one hour to obtain a result even if only a single parameter is changed. Therefore, a mid-size scenario between scenarios 1 and 2 was designed for extensive analysis of this proposal. Scenario 3 consists of 16 ECUs and 86 tasks as shown in Table III. The ECU capacity and task demand are given as 169 MB and 124 MB, respectively. The total resource capacity of all ECUs is 129% with respect to the total resource demands of all tasks.

Figure 5 shows the experimental results to obtain the total cost according to the round with no specific initialization method in scenario 3. The round is identical to the number of SA processes. Since the SA process (line 12-21 of Algorithm 1) has statistical characteristics, the experiments were conducted more than ten times. The black dots depict the average total cost with respect to the number of SA processes. The short bars connected to the dots depict the standard deviations.

In Fig. 5, all tasks were allocated to specific ECUs at a point 0. Namely, \( |L_2| = 0 \) at the point. At that moment, the objective value is 135809.31. It took 3 s to reach point 0. At point 1, the proposed solver obtained a local optimum (128923.21). Subsequently, the SA process was conducted. On average, 125815.70, i.e., 1.45% near-optimal, can be achieved within the sixth rounds.

Figure 6 shows the experimental results using the greedy initialization method, as explained in Algorithm 2. In Fig. 6, the greedy initialization method finishes at a point a (162160.16). In other words, point a is the first moment when \( |L_2| = 0 \). After the repeated matching method, another local optimum (125976.26, 1.57%) was obtained at point b. Within five more SA processes, the solver obtained solutions with an average cost of 125145.07 (0.91%). Note that the proposed solver saves the minimum solution until the end of the searching process.

Table VI summarizes the impact of the greedy method for initialization. The initialization means to make \( L_2 \) empty. The final result of the greedy method is larger than the case of no specific initialization method as much as 19.40%. The greedy method is six times faster. Furthermore, the first local optimum is as much as 2.39% less. On the other hand, the greedy method took two times when the time between this initialization and first local optimum was considered.

The implication of Table VI is that the greedy method for initialization can effectively reduce the calculation time while not having negative effects for obtaining a solution close to the global optimum.

### C. Sensitivity Analysis

Several parameters affect the performance of the proposed method, such as the ratio of tasks to ECUs (\( N/K \)), ECU installation cost (\( c_j \)), task installation cost (\( q_{ij} \)) and available resource capacity of the ECU (\( c_j \)).
Fig. 7. Total cost according to the average installation cost of ECU, in scenario 1. Regions A, B, and C correspond to six, five and four ECUs, respectively.

average ECU installation cost is between 6325.0 and 7044.5, VCN can be designed using four ECUs, if the other parameters are fixed.

This paper focused on how to reduce the huge solution space of the VCN design problem, particularly, using the proposed task-to-ECU mapping without a newly-developed solution method for the EPP. In most of modern vehicle scenarios, the complexity may not be a bottleneck for solving the VCN design problems because \( J < K < 100 \), typically. However, as VCN technologies allow more ECUs to be adopted, a new solution method needs to be developed to solve the EPP rapidly. Future work includes developing and speeding up the method.

### VIII. Conclusion

This paper reports a new methodology for the optimal VCN design. The VCN design was first defined as the temporal and spatial joint optimization problem, and some of the challenges in solving the problem were presented. To meet the challenges, an analytical model to examine the fundamental characteristics of the problem was derived. A repeated-matching-based fast solution method was next provided to optimize the VCN design. The test results showed that the proposed method can obtain 1% gap near-optimal solutions within a significantly reduced time in different real-world scenarios.

The concept of temporal and spatial joint optimization that this paper proposes can help model the inter-dependency analytically and reduce the design complexity. The fast solution methodology is expected to contribute to the decrease in design time for vehicles that should support a range of customer options and new functionalities. The proposed methodology can be applied to automated task-ECU mapping, and is expected to support the efforts to quantify the installation cost and resource profile of vehicle tasks.
APPENDIX

A. Matching Cost Calculation

Wark et al. provided the basics for calculating six blocks of the repeated matching method in [6]. This paper generalizes the method to a multi-dimensional space. In addition, the proposed model considers reliability constraint as shown in Sec. II.C. The practical implementation issues, not described in [6], is also provided in detail.

Let us investigate the methods calculating each block of a matching cost matrix $b$ in detail as follows:

1) **Block 1, assigning an unused ECU to an unused ECU:**

Matching of an unused ECU with another unused ECU is not feasible and hence the cost can be set to $\infty$ (large value in practice). Since self matching is that the ECU is unused, the given cost is 0. Let $i$ be $r$th ECU of $L_1$ and $j$ be $s$th ECU of $L_1$. Therefore, the matching cost is

$$b_{r,s} = \begin{cases} 
\infty & \text{if } r \neq s \\
0 & \text{otherwise}.
\end{cases}$$

In Fig. 2, $L_1 = \{E_2\}$. Since $E_2$ is the first ECU of $L_1$, $r = s = 1$ and $b_{1,1} = 0$. In this case, the system can reduce the weight and volume as much as the $E_2$ had and occupied.

2) **Block 2, assigning an unassigned task to an unused ECU:**

Matching of an unassigned task with an unused ECU is feasible, if the capacity of the ECU is not exceeded by the resource demand of the task. Let $i$ be $s$th ECU of $L_1$ and $j$ be $r$th task of $L_2$. The matching cost of Block 2 is

$$b_{n_1+r,s} = \begin{cases} 
eq c_i & \text{if } s_j \leq c_i \\
\infty & \text{otherwise}.
\end{cases}$$

Fig. 2 shows that $T_2$ is assigned to $E_2$ if the resource demand of $T_2$ is less than the capacity of $E_2$.

3) **Block 3, assigning an unassigned task to an unassigned task:**

Matching of an unassigned task with another unassigned task is not possible and hence the cost is $\infty$. Self-matching means the task is not assigned to any ECU, which means the task cannot work. Since this situation is not allowed in practice, the given cost of this method should be a sufficiently large value, $W_{\infty}$. Let $r$ be the $i$th task of $L_2$ and $s$ be the $j$th task of $L_2$. The matching cost of Block 3 can be represented as

$$b_{n_1+r,n_1+s} = \begin{cases} \infty & \text{if } r \neq s \\
2W_{\infty} & \text{if } r = s.
\end{cases}$$

$2W_{\infty}$ represents the summation of both $r$'s $W_{\infty}$ and $s$'s $W_{\infty}$. This suggests that each task has to be assigned to at least one ECU. In Fig. 2, $T_3$ incurs high cost. Note that $b_{ij}$ must be double the actual cost in the case of self-matching ($i = j$).

Blocks 1, 2, and 3 are intuitive to be calculated, but Block 6 is not straightforward. Block 6 is examined first because Blocks 4 and 5 are special cases of Block 6.

4) **Block 6, rearranging two used ECUs:**

The goal of Block 6 is to calculate the minimum cost matching between two ECUs having more than one tasks. In Fig. 2, assigning $T_3$, $T_6$ and $T_7$ to $E_3$ and $E_4$ is an example of Block 6. To find the minimum cost re-packing two elements $(j_1, R_1), (j_2, R_2) \in L_3$, the matching was split into three cases. 1) $R_1$ and $R_2$ are assigned to $j_1$, if the total demand of $R_1$ and $R_2$ does not exceed the resource capacity of $j_1$. 2) $R_1$ and $R_2$ are assigned to $j_2$, if the total demand of $R_1$ and $R_2$ is less than the capacity of $j_2$. 3) $R_1$ is assigned to $j_2$ and $R_2$ is assigned to $j_1$, i.e., swapping of tasks between the two ECUs but both ECUs are still in use. Cases 1 and 2 can be checked easily by comparing the total demand from the tasks with the capacity of each ECU. However, it is not straightforward to solve case 3 because the best swapping of tasks between ECUs needs to be found. This is also an ILP problem. Since a task $t \in \{R_1 \cup R_2\}$ can be allocated to either $E_{j_1}$ or $E_{j_2}$, the computation complexity for the case 3 solution is $O(2^{|R_1|+|R_2|})$. To solve the problem within the polynomial time, case 3 is reformulated into a 0-1 knapsack problem. Let us introduce the following variables:

$$W_j = \begin{cases} 1 & \text{if task } i \in R_1 \text{ swaps to ECU } j_2 \\
0 & \text{otherwise}
\end{cases}$$

$$Y_i = \begin{cases} 1 & \text{if task } i \in R_2 \text{ swaps to ECU } j_1 \\
0 & \text{otherwise}
\end{cases}$$

With these variables, the swapping problem can be formulated as

$$\begin{align*}
\min & \quad o^* = \sum_{i \in R_1} V_i W_i + \sum_{i \in R_2} X_i Y_i \\
\text{s.t.} & \quad -\delta_y \preceq \sum_{i \in R_1} -s_i W_i + \sum_{i \in R_2} s_i Y_i \preceq \delta_w (20) \\
& \quad W_i, Y_i \in \{0, 1\} \quad \forall i.
\end{align*}$$

where $o^*$ means a marginal cost by reassigning tasks ($\in \{R_1 \cup R_2\}$) into $E_{j_1}$ and $E_{j_2}$. $V_i$ means a marginal cost if task $i$ swaps from $E_{j_1}$ to $E_{j_2}$, i.e., $V_i = q_{i,j_2} - q_{i,j_1}$. Similarly, $X_i = q_{i,j_1} - q_{i,j_2}$. The decision variables are $W_i$ and $Y_i$. $-s_i W_i + \sum_{i \in R_2} s_i Y_i$ means the increased resource demand w.r.t. $E_{j_1}$ after swapping with $[W_1, \ldots, W_{|R_1|}, 1, \ldots, Y_{|R_2|}]$. $\delta_w$ means the available capacity of $E_{j_2}$, i.e., $\delta_w = c_{j_2} - \sum_{i \in R_1} s_i$. Similarly, $\delta_y = c_{j_1} - \sum_{i \in R_1} s_i$. (20) can be solved using a dynamic programming method with the computation complexity with computation complexity $O(\hat{R} \times \hat{S})$, where $\hat{R} = |R_1| + |R_2|$, $\hat{S} = \max s_i, i \in \{R_1 \cup R_2\}$. The reformulation can be written as

$$\begin{align*}
\max & \quad o^* = -\sum_{i \in R_1} V_i W_i - \sum_{i \in R_2} X_i Y_i \\
\text{s.t.} & \quad -\delta_y \preceq \sum_{i \in R_1} -s_i W_i + \sum_{i \in R_2} s_i Y_i \preceq \delta_w (21) \\
& \quad W_i, Y_i \in \{0, 1\} \quad \forall i.
\end{align*}$$

where $R_i^-$ denotes the set of tasks with negative $V_i$, $R_i^- \subset R_i$. Similarly, $R_i^+$ is the set of tasks with negative $X_i$, $R_i^+ \subset R_i$. The computation complexity of (22) is $O(R^- \times S^*)$, where $R^+ = |R_1^-| + |R_2^+|$, $S^* = \max s_i, i \in \{R_1^- \cup R_2^+\}$. Note that $o^* = -o^*$. In the case of a large number of ECUs involved, the problem can be solved using a number of different heuristic methods, which have been proposed for the knapsack problem. However, since only two ECUs are involved in (21), (21) can be solved rapidly even with an ILP solver. To find the matching cost, we simply compare the costs corresponding to the three cases. Let
\((j_1, R_1)\) and \((j_2, R_2)\) be the \(r\)th and \(s\)th elements in the set \(L_3\), then
\[
b_{n_1+n_2+r,n_1+n_2+s} = \begin{cases} 2\left( e_{j_1} + \sum_{i \in R_1} q_{ij_1} \right) & \text{if } r = s, \\ \min\{o_1, o_2, o_3\} & \text{otherwise} \end{cases}
\]
where
\[
o_1 = \begin{cases} e_{j_1} + \sum_{i \in R_1} q_{ij_1} + \sum_{i \in R_2} q_{ij_2} & \text{if } \sum_{i \in R_1} s_i + \sum_{i \in R_2} s_j \leq c_{j_1}, \\ \infty & \text{otherwise} \end{cases}
\]
\[
o_2 = \begin{cases} e_{j_2} + \sum_{i \in R_1} q_{ij_2} + \sum_{i \in R_2} q_{ij_2} & \text{if } \sum_{i \in R_1} s_i + \sum_{i \in R_2} s_j \leq c_{j_2}, \\ \infty & \text{otherwise} \end{cases}
\]
\[
o_3 = e_{j_1} + \sum_{i \in R_2} q_{ij_2} + e_{j_2} + \sum_{i \in R_2} q_{ij_2} + o^*.
\]
\(o_1\) and \(o_2\) correspond to the two cases where one ECU is allocated to all tasks and the others become unused. \(o_3\) is the case when both ECUs are in use but there is a swapping of tasks between them. In the case of no swapping, the original cost is kept intact because \(o^*\) is zero.

5) **Block 4, rearranging a used ECU and an unused ECU:**
The goal of Block 4 is to calculate minimum cost matching between the ECU assigned tasks and the ECU assigned no task. In Fig. 2, assigning \(T_3\) and \(T_7\) to \(E_2\) and \(E_3\) is an example of Block 4. To match an open ECU with the assigned tasks with an empty ECU is a special case of Block 6. The difference is that the number of tasks assigned to the unused ECU is zero and the surplus capacity equals the overall capacity of the unused ECU. To use the problem and simplify the description, an empty ECU can be described as an element \((j_2, R_2)\). The following modification should then be applied to (21). Let \(j \in L_1, \) set \(R_2 = \emptyset\) and \(\delta_2 = c_{j_2}\). To determine the matching cost, let \(j\) be the \(r\)th ECU of \(L_1\), and \(i\) be the \(s\)th element in \(L_3\). Finally, the cost is as follows:
\[
b_{n_1+n_2+r,s} = \min\{o_1, o_2, o_3\}
\]
where \(o_1, o_2\) and \(o_3\) are given by (23), (24), and (24).

6) **Block 5, assigning an unassigned task to an used ECU:**
The goal of Block 5 is to assign unassigned tasks to an ECU already assigned tasks. In Fig. 2, how to assign \(T_5, T_6\) and \(T_7\) to \(E_2, E_3\) is an example of Block 5. This is not straightforward because there can be a number of unassigned tasks due to the capacity limitation of \(E_3\). There are two cases when a used ECU, i.e., ECU in \(L_3\), is matched to an unassigned task. The first is that the unassigned task can be assigned to the ECU, i.e., the capacity limitation is not exceeded by adding a new task. If the capacity is exceeded, one or more tasks must be unassigned. Let \((j_1, R_1)\) be the \(r\)th element in \(L_3\) and \(i\) be the \(s\)th element in \(L_2\). The block 5 problem can be formulated as follows:
\[
\begin{align*}
\min f_o(R_1, j_1, t) &= \sum_{i \in \{v \in R_1\}} q_{ij_1} Q_i + |\pi_t| W_\infty \\
\text{s.t.} & \sum_{i \in \{v \in R_1\}} s_i \hat{Q}_i \leq c_{j_1}
\end{align*}
\]
where \(Q_i\) is a binary variable equal to 1 if \(T_i\) is allocated to \(E_{j_1}\), 0 otherwise. \(Q_i\) is a decision variable. \(Q_i\) is the identity matrix if \(Q_i = 1\), otherwise \(Q_i\) is zero matrix. \(\pi_t\) denotes the number of unassigned tasks among \(\{t \cup R_1\}\). As a result, the matching cost of the block 5 is
\[
b_{n_1+n_2+r,s} = e_{j_1} + f_o(R_1, j_1, t).
\]
Note that unassigned tasks need to be included in \(L_2\). Block 5 can be solved using the solution of block 6, if \(R_2 = \emptyset\), \(R_1 = R_1 \cup t, \delta_2 = -s_i, k_i = W_\infty - q_{ij_1}, i \in R_1\), and \(k_i|_{R_1+1} = W_\infty - q_{ij_1}\).

**B. Proof of Theorem 1**

**Assumption 1:** \(\varepsilon_j < \infty, q_{ij} < \infty \quad \forall i \in T, \forall j \in T\).
Assumption 1 is feasible, because all physical installation costs are finite.

**Lemma 3:** Given a set of \(e_j, q_{ij}\) and \(\mathbb{E}\),
\[
f(\mathbb{E}) < \infty.
\]
**Proof:** By Assumption 1, \(e_j\) and \(q_{ij}\) are finite and \(y_{ij} = 1\). By the definition of \(f(\mathbb{E})\) in (1), since \(f(\mathbb{E})\) is a summation of \(e_j, q_{ij}\) and \(\mathbb{E}\), \(f(\mathbb{E})\) is finite.

Given \(e_j\) and \(q_{ij}\), \(L_1, L_2\) and \(L_3\) can be derived only with a set of decision variables \(y_{ij}, \ i \in T, j \in E\). \(x_j\) is decided only by \(y_{ij}\). \(b\) matrix is derived by solving (19) with \(L_1, L_2\) and \(L_3\). The left term of (21) represents the solving results of the right term of (21). In a matrix \(b\), only blocks 1, 3, and 6 can contain diagonal elements. The diagonal elements reside in the place where the row and column are identical. In the case of block 6, diagonal elements represent self-matching, namely no change happens. If \(b\) consists of only block 6, the left term and right term of (21) is identical because no mapping is changed. In case of block 1, self-matching incurs \(\infty\). In this case, the left term of (21) can be \(\infty\) but the right term of (21) is always finite according to Lemma 3.

**C. Derivation of CAN Timing Constraints**

Bitwise arbitration needs to be executed to determine one message that can access the CAN bus. When a certain message is broadcast on the bus, the signal arrival time at each ECU varies according to the distance between the signal source and each destination. Therefore, the delay must be considered to guarantee the proper functionality of bitwise arbitration.

In particular, a bitwise arbitration signal must arrive before a sampling point. However, a signal delivers with a delay including the processing delay and transmission delay due to wiring length and transmission rate. Hence, the sampling point must be larger than at least twice the signal delay. The sampling point \(t_{sp}\) can be formulated as
\[
t_{sp} \geq 2(t_d + t_{bus})
\]
where \(t_d = 2t_{can} + t_{tx} + t_{rx}\) is a total transmission delay of a signal. \(t_{can}, t_{tx}\) and \(t_{rx}\) denote the processor delay, transmitter delay and receiver delay, respectively. \(t_{bus} = U_{bus}/v_{bus}\) where \(U_{bus}\) is the maximum length of the bus line and \(v_{bus}\) is the speed of an electron (\(\approx 2 \times 10^8\) m/s). Since a receiver samples...
the bus at $t_{bp}$ among one bit time, $t_{bp}$ can be represented by a fraction of bit time like $t_{bp} = \delta t_{bit}$, where $0 < \delta < 1$. $t_{bit}$ is the bit transmission time, i.e., $t_{bit} = 1/b_{bus}$, where $b_{bus}$ is the baud rate.

$$t_{bp} \leq \frac{t_{bp}}{2} - \tau_d.$$  

Because, $U_{bus} \leq G_{init} + \sum_{i \in E} \sum_{j \in C} \sigma_j m_i k_{ij}$,

$$l_v + l_w + G_{init} + \sum_{i \in E} \sum_{j \in C} \sigma_j m_i k_{ij} \geq v_{bus} \left( \frac{\delta}{2b_{bus}} - t_d \right).$$

where $v, w \in \bar{O} \subset O$. $l_j$ denotes the stub-line length of an ECU located in location $j \in O$.

REFERENCES


Seongwoo Kim received the B.S. and M.S. degrees in electronics engineering from Korea University, Seoul, Korea, in 2005 and 2007, respectively, and the Ph.D. degree in electrical engineering and computer science from Seoul National University in 2011. He was a Researcher with the Intelligent Vehicle IT Research Center, which is funded by the Korean Government and automotive industries. He is currently with the Department of Electrical Engineering and Computer Science, Seoul National University. He is the author or a coauthor of more than 100 magazine articles on software engineering, security, humanoids, digital gadgets, and technical writing. His research interests include on-line and off-line optimization targeted for autonomous vehicle, security, smart grid and computer networks.

Dr. Kim received the Best Student Paper Award at the 11th IEEE International Symposium on Consumer Electronics and the Outstanding Student Paper Award at the First IEEE International Conference on Wireless Communication, Vehicular Technology, Information Theory, and Aerospace and Electronic Systems Technology.

Eundong Lee received the B.S. degree in electrical engineering from the Korea Advanced Institute of Science and Technology, Daejeon, Korea, in 2007 and the M.S. degree in electrical engineering from Seoul National University, Seoul, Korea, in 2009. He is currently working toward the Ph.D. degree with the Department of Electrical Engineering and Computer Science, Seoul National University. He is currently a Researcher with the Intelligent Vehicle IT Research Center, which is funded by the Korean Government and automotive industries. His research interests include industrial control networks and electrical power systems.

Mideum Choi received the B.S. degree in electrical engineering from Korea University, Seoul, Korea, in 2008. He is currently working toward the Ph.D. degree with the Department of Electrical Engineering and Computer Science, Seoul National University. He is also currently a Researcher with the Intelligent Vehicle Research Center, which is funded by the Korean Government and automotive industries. His research interests include hybrid energy storage systems and battery management.

Hanyou Jeong received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer science from Seoul National University, Seoul, Korea, in 1998, 2000, and 2005, respectively. From 2005 to 2007, he was a Senior Engineer with the Telecommunication R&D Center, Samsung Electronics, Suwon, Korea. In 2008, he joined the Digital Technology Center, University of Minnesota, Minneapolis, as a Postdoctoral Research Fellow. He is currently an Assistant Professor with the Institute of Logistics Information Technology, Pusan National University, Busan, Korea. His research interests include wireless networks, vehicular ad hoc networks, and optical networks.

Seongwoo Seo (M’97) received the B.S. and M.S. degrees in electrical engineering from Seoul National University, Seoul, Korea, and the Ph.D. degree in electrical engineering from Pennsylvania State University. He was with the faculty of the Department of Computer Science and Engineering, Pennsylvania State University, and served as a Member of the Research Staff with the Department of Electrical Engineering, Princeton University, Princeton, NJ. In 1996, he joined the faculty of Seoul National University, where he served as the Director of the Information Security Center for five years. He is currently a Professor of electrical engineering with the Department of Electrical Engineering and Computer Science, Seoul National University, and the Director of Intelligent Vehicle Research Center, which is funded by the Korean Government and automotive industries. His research areas include vehicular electronics and communication networks, network security, and system optimization.

Dr. Seo has served as the Chair or a Committee Member for several international conferences and workshops, including the IEEE International Conference on Computer Communications, the IEEE Global Communications Conference, the IEEE Symposium on Personal, Indoor, Mobile, and Radio Communications, the IEEE Vehicular Technology Conference, the ICST Conference on Security and Privacy in Mobile Information and Communication Systems, and the IEEE International Conference on Wireless Communication, Vehicular Technology, Information Theory, and Aerospace and Electronic Systems Technology.