A Genetic Algorithm for Packing CAN FD Frame with Real-Time Constraints

Shan DING†a, Gang ZENG†b, Nonmembers, Ryo KURACHI††c, Member, and Ruifeng HUANG†, Nonmember

SUMMARY As a next-generation CAN (Controller Area Network), CAN FD (CAN with flexible data rate) has attracted much attention recently. However, how to use the improved bus bandwidth efficiently in CAN FD is still an issue. Contrasting with existing methods using greedy approximate algorithms, this paper proposes a genetic algorithm for CAN FD frame packing. It tries to minimize the bandwidth utilization by considering the different periods of signals when packing them in the same frame. Moreover, it also checks the schedulability of packed frames to guarantee the real-time constraints of each frame and proposed a merging algorithm to improve the schedulability for signal set with high bus load. Experimental results validate that the proposed algorithm can achieve significantly less bandwidth utilization and improved schedulability than existing methods for a given set of signals.

key words: in-vehicle network, Controller Area Network, frame packing, network bandwidth, schedulability

1. Introduction

In recent years, the electronic control systems and in-vehicle networks of automobiles have become much more sophisticated and complex. A modern automobile may have more than 100 electronic control units (ECUs) for various subsystems. These subsystems are connected by the in-vehicle network to achieve integrated control for fuel efficiency and active safety. Currently, the most widely used in-vehicle network is the Controller Area Network (CAN) [11].

With the increased communication requirements, CAN has reached its bandwidth limits in some application fields. To improve the bandwidth of CAN, Bosch has proposed an improved CAN data link layer protocol, called CAN with flexible data rate (CAN FD). There are two significant improvements comparing with standard CAN. One is the increased data bit rate. The data bits can be transferred with up to 8Mbps while the standard CAN can only transfer with maximum 1Mbps. The other is the increased payload size. The CAN-FD frame allows up to 64 bytes payload compared with the maximum of 8 bytes with standard CAN. While the CAN FD can improve the bandwidth of CAN, there are some challenges that must be addressed to take advantage of it. A main issue is how to pack various signals with different periods and deadlines into CAN FD frames to minimize the bandwidth utilization and meet their deadlines simultaneously. There are several difficulties to solve the issue. One is that packing signals with different periods into one frame could lead to waste of bandwidth. The other is that all packed frames should meet their deadlines when transmitting them on the CAN bus. Additionally, as stated in [2], the frame packing problem is a typical variable sized bin packing problem, which is NP hard.

There are some existing methods to deal with the frame packing problem in the literature. Sandstrom et al. [9] proposed a next-fit decreasing based heuristic for the classic bin packing problem. The algorithm sorts the signals by deadlines and then iteratively assigns a signal to a frame. In case a signal cannot fit into existing frame anymore, a new frame is created in the next iteration. In contrast, Saket and Navet [8] sort the signals by their bandwidth utilization and try to pack the signals with similar periods in the same frame. Polzbauer et al. [6] also presented a heuristic based on the next-fit decreasing algorithm, and proposed an additional standard for assessing whether to add a signal to an existing frame according to its impact on bandwidth utilization. Recently, Bordoloi and Samii [2] proposed a two-stage framework for CAN FD Frame Selection (CaFeS) problem and schedulability verification of packed frames. In the first stage, a dynamic programming method was proposed for bandwidth optimized packing subject to the CAN FD size requirement. In the second stage, the signals with missed deadline will be unpacked firstly, and finally be packed using next-fit heuristic.

Although all existing methods are effective for CAN FD frame packing, there is room for further improvement in terms of bandwidth minimization and schedulability. Different from existing methods which used greedy approximate algorithms to solve the NP hard problem, we propose an evolutionary algorithm, i.e. a genetic algorithm to solve it. The proposed GA is dedicated for frame packing with special considerations on the different period of signals and the deadline constraint of each signal. Moreover, a merging algorithm is proposed to improve the schedulability of packed frames for signal set with high bandwidth utilization. The experimental results on randomly generated signal sets showed significant reduction of bandwidth utilization and improvement of schedulability over the state-of-the-art method.
The rest of the paper is organized as follows. Section 2 gives an overview of the CAN FD and frame packing problem. In Sect. 3, the proposed genetic algorithm is presented. Section 4 describes experimental results, and finally Sect. 5 concludes the paper.

2. CAN FD Frame Format and Frame Packing Problem

In this section, we first introduce the CAN FD frame format, and present the formulation of frame packing problem.

2.1 CAN FD Frame Format and Main Difference with CAN

Figure 1 depicts the frame format of CAN FD, which includes arbitration phase and data phase. The main differences between CAN FD and CAN 2.0 are as follows. (1) A larger payload size is available. The possible payload sizes are 8, 12, 16, 20, 24, 32, 48, and 64 bytes in contrast to the maximum of 8 bytes in CAN 2.0. The larger payload size allows to pack more signals in a frame. (2) While the arbitration phase has the same bit rate to CAN 2.0, the data phase can be transferred with up to 8Mpbs bit rate which depend on CAN transceiver signal delay and the signal delay on the cable. (3) There are three new bits in the control field of CAN FD as shown in Fig. 1. The EDL (Extended Data Length) bit is used to distinguish between the standard CAN frame and the CAN FD frame. The value of BRS (Bit Rate Switch) indicates whether the bit rate of data phase is the same as the arbitration phase or a predefined faster bit rate is used in the data phase. The ESI (Error State Indicator) bit indicates the transmitter’s error state.

2.2 Worst Case Transmission Time Calculation

As introduced in [2], we also adopt Eq. (1) to calculate the worst case transmission time (WCTT) of a CAN FD frame in the worst-case bit stuffing scenario.

\[
WCTT(p) = 32t_{arb} + \left(28 + 5\left\lceil \frac{p-16}{64} \right\rceil + 10p\right)t_{data}.
\]

where \( p = 0, 1, \ldots, 8 \) for CAN 2.0 or \( p = 12, 16, 20, 24, 32, 48, 64 \) for CAN FD, and \( t_{arb} \) and \( t_{data} \) represent the time to transmit one bit during the arbitration phase and data phase, respectively. For example, if the arbitration bit-rate is 500 kbps and the data bit-rate is 2 Mbps, then \( t_{arb} = 2 \mu s \) and \( t_{data} = 0.5 \mu s \).

Note that because the arbitration bits are fixed, the larger the payload of a frame is, the smaller the relative overhead for the frame transmission is. Therefore, in general, trying to pack as many signals in a frame may reduce the bandwidth utilization of CAN FD if all signals have the same periods.

2.3 Formulation of Frame Packing Problem

The frame packing problem can be described as follows: for a given set of signals, try to pack them in a set of frames such that the used bandwidth of CAN FD is minimized and all packed frames can meet their deadlines. The formulation of the problem is given below.

- **Input**: a given signal set \( S \), and each signal \( s_i \) can be characterized by \( [h_i, d_i, l_i] \), which represents period \( h_i \), deadline \( d_i \), and size (payload) \( l_i \) of the signal, respectively.
- **Output**: a set of packed frames containing all input signals. A frame \( f_j \) consists of a set of signals \( S_j \), where \( S_j \) is a subset of \( S \). Note that signals from two different ECUs are not allowed to be packed into the same frame. A frame \( f_j \) can be characterized by \( [H_j, D_j, L_j] \) which denote the period \( H_j \), deadline \( D_j \), and size \( L_j \) of a frame, respectively. Specifically, the deadline \( D_j \) of a packed frame is the minimum deadline among the deadlines of packed signals in the same frame [2]. For example, if three signals with deadlines of 10ms, 15ms, and 20ms are packed in a frame, the deadline of the frame should be the minimum deadline i.e., 10ms to meet all deadlines of three signals. The size \( L_j \) of a frame is the summation of the payload of all packed signals in the same frame. Once the size of a frame is known, we can use Eq. (1) to calculate its worst-case transmission time. Note that a frame is associated with a priority identifier \( ID_j \) that uniquely identifies the priority of the frame for purposes of arbitration on the CAN bus.

- **Objective**: minimizing the bandwidth utilization of CAN FD by packing a given signal set in frames. The objective can be formulated as follows.

\[
\text{Minimize } B
\]

The bandwidth utilization, \( B \), of a set of packed signals can be calculated by the following equation.

\[
B = \sum_{i=1}^{F} \frac{WCTT(W^i_k)}{H_i}, \quad f_i \leq W^i_k,
\]

where \( F \) is a set of packed CAN FD frames, and \( f_i \) is the payload size of the \( i \)th packed frame. \( W^i_k \) is the CAN FD frame size of the \( k \)th type of frame corresponding to a packed frame \( f_i \). CAN FD allows payloads of 1, 2, 3, 4, 5, 6, 7, 8, 12, 16, 20, 24, 32, 48 or 64 bytes. Therefore, \( W^i_k \) refers to one value of the above bytes, and \( k \in [1, 15] \). \( H_i \) is the period of the \( i \)th frame, which can be calculated as follows.

\[
H = \min_{i=1}^{F} \{h_i\},
\]
where \( h_j \) is the period of the \( j \)th signal packed in the \( i \)th frame.

- **Constraints:** There are two constraints for the optimal frame packing problem. One is the size limitation for each selected frame type. The other is the real-time constraint, that is, all frames should meet their deadlines.

Hardness: As described earlier, there are three challenges for the CAN FD frame packing problem. First is the variable sized bin packing problem that is \( NP \) hard. Second is the effect of different periods of signals on bandwidth utilization. While packing signals with the same periods in a frame can save bandwidth, packing signals with different periods in a frame may waste bandwidth. Third is the real-time constraint. The packed frames with minimum bandwidth may be infeasible if they cannot meet all deadlines of the frames. In this paper, we will aim at these issues, and propose a dedicated genetic algorithm to solve them.

3. Genetic Algorithm for Frame Packing

Genetic Algorithm (GA) is a search heuristic that is used to solve the optimization problems in the field of computer science and artificial intelligence [1]. The frame packing problem is a kind of optimization problems that are suitable for GA to solve. An overview of the proposed GA is shown in Fig. 2, where the schedulability checking is an added step for the frame packing problem. In order to apply the general GA, we define a set of genetic operations and an evaluation function dedicated to the frame packing problem in the following sections.

3.1 Generate Initial Generations

The first step of genetic algorithm is to generate a set of candidate individuals (i.e., the initial generation). Since our purpose is to find an optimal packing of the signals, the individuals are a set of packed frames containing all signals. The CAN FD frame packing needs to consider the size, period and deadline of both signals and frames. These parameters can be calculated as described in the Sect. 2.3. In this section, we introduce how to create an individual as the initial generation.

A set of signals is given as an input for packing, which is denoted by \( S = \{s_1, s_2, \ldots, s_N\} \), where \( N \) is the number of signals. The output is the individual that contains a set of frames, and each frame consists of a subset of packed signals of \( S \). In this paper, an array is employed to represent the individual (i.e., a set of packed signals). As shown in Fig. 3, the index (label) of the array represents the number of the signal, and the contents of the array indicate which frame the signal is packed in.

**Example:** Considering ten signals \( S = \{s_1, s_2, \ldots, s_{10}\} \), \( N = 10 \), have been packed in three frames, denoted by \( f_1 = \{s_1, s_7, s_9\} \), \( f_2 = \{s_1, s_2, s_8\} \), \( f_3 = \{s_5, s_8, s_{10}\} \), respectively. Therefore, the individual can be represented by the array in Fig. 3.

Next, we describe the detailed steps to generate the individuals as the initial generations. (1) First, to find out the global optimal solution, ensuring the randomness of the generated individual is critical in this step. To this end, a candidate individual can be obtained by the following iterative operations. Given a signal set \( S \) for packing, \( S_i \) is defined as the set of unpacked signals of \( S \) at iteration \( i \). At the first iteration, \( i = 1 \) and \( S_1 = S \). At next iteration, the following procedure is conducted to generate a packed frame \( f_i \) randomly. A random number \( t_{ni} \) between 1 and \( n_i \) is generated indicating \( t_{ni} \) numbers of signals will be packed in frame \( f_i \), where \( n_i \) denotes the total number of unpacked signals. Then random numbers representing the remaining unpacked signals will be generated \( t_{ni} \) times, and a set of random numbers is obtained which is denoted by \( T = \{t_1, t_2, t_3, \ldots, t_{ni}\} \). Every \( t \) in \( T \) represents one unpacked signal that should be packed into the same frame. Finally, a frame \( f_i \) is produced. (2) Second, checking the randomly generated frame \( f_i \), to see if it satisfies the following constraints of CAN FD frame.

- **Size constraint:** In addition to the standard CAN sizes, CAN FD allows payloads of 12, 16, 20, 24, 32, 48 or 64 bytes. Therefore, the accumulated size of signals in one frame should be equal or less than the above limitation.
- **Real-time constraint:** Each frame has a deadline, we need to calculate the worst case transmission time on the CAN bus, to see if its WCTT is less than or equal to
its deadline. Note that we only evaluate the worst case transmission time but not the worst case response time of the frame when generating the initial generation. A main reason is that the initial generation that can meet deadlines cannot always lead to better result than an initial generation with both schedulable individuals and unschedulable individuals. In other words, keeping the diversity of the initial generation is important to obtain qualified solution after evaluation.

If the randomly packed frame fails to meet the above two constraints, we should generate it again. If the frame is feasible, the packed signals denoted by $S^f$ are removed from $S_i$ to obtain $S_{i+1}$. This process is iterated until all signals have been packed. When the iterations terminate, a set of packed CAN FD frames are generated denoted by $F = \{f_1, f_2, \ldots, f_N\}$. In the same way, any number of individuals can be obtained as the operation flow shown in the Fig. 4.

**Example**: Consider the individual in Fig. 3 as an example. There are ten signals, $S = \{s_1, s_2, \ldots, s_{10}\}$, $N = 10$. Therefore $S_i = S$ and $n_i = 10$. (1) First, a random number (e.g., $t_{n_1} = 3$) between 1 and 10 is generated, which means that three signals will be packed into the frame $f_1$. Accordingly, we generate 3 random numbers (e.g., 4, 7, 9) between 1 and 10 (denoted by $T = \{4, 7, 9\}$), which indicate that three signals, i.e., $s_4$, $s_7$, and $s_9$ will be selected to pack in frame $f_1$. As a result, the packed frame $f_1 = \{s_4, s_7, s_9\}$ is generated. (2) Second, checking the packed frame $f_1$, to see if it satisfies the constraints of CAN FD. In case that it is feasible, the packed signals denoted by $S^f = \{s_4, s_7, s_9\}$ are removed from $S_i$, and the remaining signal set is updated to $S_2 = \{s_1, s_2, s_3, \ldots, s_9\}$. (3) In the second iteration, a random number (e.g., $t_{n_2} = 3$) between 1 and 7 is generated, which means that three signals will be packed into the frame $f_2$. Next, 3 random numbers (e.g., 1, 2, 5) corresponding to the indexes of unpacked signals in $S_2$ are generated, which indicates that three signals, i.e., $s_1$, $s_2$, and $s_6$ will be selected to pack in frame $f_2$. (4) Repeat the above procedure, three packed frames e.g., $f_1 = \{s_4, s_7, s_9\}$, $f_2 = \{s_1, s_2, s_6\}$, and $f_3 = \{s_3, s_5, s_8, s_{10}\}$ can be obtained, finally.

### 3.2 Fitness Calculation

GA is a search heuristic which mimics the process of natural selection. After the initial population is generated, the better individuals should be left according to the principle of survival of the fittest. Intuitively, the bandwidth utilization of each individual can be used as one of the fitness metrics because our objective is to minimize it.

In general, the lower bandwidth utilization the individual has, the better fitness it will have. Specifically, the bandwidth utilization of an individual can be calculated using the Eq. (2). After calculating the fitness of each individual, a standard generation can be selected by sorting the individuals in the descending order of their fitness [3]. We defined $N_{ind}$ as the number of the individuals in one generation, and $N_{gen}$ as the number of the generations, to control the entire evolution process [4].

As mentioned early, because all packed frames must meet their deadlines when transmitting them on the CAN bus, additional fitness metric should be introduced. That is, schedulability of frames, which is denoted by $Sch$. If all packed frames of an individual can meet their deadlines, $Sch = 0$, otherwise $Sch = 1$. Schedulability test is introduced in Sect. 3.6. Therefore, the overall fitness of an individual, $i$, can be calculated by:

$$Fitness(i) = \frac{1}{Sch(i) + B(i)}, \quad (4)$$

where $Sch(i)$ is 1 or 0, and $B(i)$ is a value between 0 and 1. In general, the higher the fitness is, the better the packed result is. Note that even though the bandwidth utilization is small, the fitness of a generation will be less than 1 if the packed frames cannot meet their deadlines. In other words, the metric gives the schedulability larger weight than the bandwidth utilization. The detailed description about schedulability test will be given in Sect. 3.6.

### 3.3 Selection Operation

For selection operation, first, all individuals in one generation are sorted according to decreasing fitness value, and the first individual will be the best packed result in this generation. Second, the following selection operation is performed to obtain the next generation as shown in Fig. 5 with an example.

To reserve more excellent individuals in the next generation, we perform crossover operation (details will be given in the next section) by choosing the $i$th individual and its next individual of the previous generation as the parents, where $i = 1, 2, \ldots, \lceil p/2 \rceil$ and $p$ is the population size. This operation expects to generate a new generation with higher fitness because the fitness of their parents is high. In addition, we also perform crossover operation by choosing the
The previous generation

The next generation

Fig. 5 An example of selection operation

The jth individual and the \((p + 1 - j)\)th individual of the previous generation as the parents, where \(j = 1, 2, \ldots, [p/2]\) [5]. This operation is to maintain the continuity of individuals, which is also important to keep both good and bad individuals coexisting in the next generation.

3.4 Crossover Operation

GA simulates the natural genetics, and performs crossover operation on genetic operators. This process will result in the enhancement of the ability to adapt to the environment. The best individual of the last generation can be used as the approximate optimal solution for the CAN FD packing problem.

The crossover operation is conducted between parents. In this study, we adopt the binary crossover, which is comprised of two steps as follows.

- Step1: The one-point crossover [6] is performed at the crossing point \(k \in \{1, N - 1\}\), and partial array variables of the parents are exchanged mutually. After the crossover operation, two individuals are obtained as the children.
- Step2: The following two conditions are applied one by one to select only one individual as the new generation.

Condition 1: If the majority of all signals have the same period, only the individual with less number of frames is chosen as the next generation.

Condition 2: If the periods of the signals are different from each other, only the individual that has more number of signals with the same periods is chosen as the next generation.

**Example:** Consider the individuals in Fig. 6 as an example. There are ten signals, \(S = \{s_1, s_2, \ldots, s_{10}\}\), and \(H = \{h_1, h_2, \ldots, h_{10}\}\), are the periods of signals. After crossover operation, two children are generated as follows. Child 1 (denoted by \(C_1 = \{f_1, f_2, f_3, f_4\}\)) consists of four frames: \(f_1 = \{s_1, s_9\}\), \(f_2 = \{s_1, s_2, s_3, s_4\}\), \(f_3 = \{s_3, s_4, s_7\}\) and \(f_4 = \{s_{10}\}\). Child 2 (denoted by \(C_2 = \{f_1, f_2, f_3\}\)) consists of three frames: \(f_1 = \{s_1, s_3, s_7, s_9\}\), \(f_2 = \{s_6\}\) and \(f_3 = \{s_1, s_2, s_4, s_8, s_{10}\}\).

Case 1: \(h_1 = h_2 = \ldots = h_{10}\). In this case, all signals have the same periods, the \(C_2\) is selected as the next generation according to the first condition because \(C_1\) has four frames while \(C_2\) only has three frames.

Case 2: The periods of the signals are different from each other, for example, \(H = \{4, 3, 1, 4, 4, 3, 4, 5\}\). In this case, the period of each signal in the packed frames should be checked. As a result, \(C_1\) will be reserved according to the second condition because \(C_1\) has 5 signals with the same periods while \(C_2\) only has 4 signals with the same periods. The signals in the same packed frame having the same periods are denoted with red mark below.

![Fig. 6 An example of crossover operation](image)

3.5 Mutation Operation

To prevent the premature convergence of genetic algorithm [7], we introduce the mutation operation as follows.

A probability, denoted by \(p_m\), is defined, to determine whether the mutation occurs or not for each individual in one generation. If the mutation occurs, we randomly find out a frame from an individual of one generation and unpack this frame by removing the signals that belong to the frame to other frames randomly.

**Example:** Consider the individual in Fig. 3 as an example. There are three frames denoted by \(f_1 = \{s_4, s_7, s_9\}\), \(f_2 = \{s_5, s_2, s_6\}\), and \(f_3 = \{s_3, s_5, s_8, s_{10}\}\). If the mutation occurs, one of the frames, such as \(f_1\), will be selected randomly. Then, the signals in \(f_1\) should be removed and repacked into other frames randomly. For example, \(s_4\) and \(s_7\) are repacked into the frame \(f_2\), and \(s_5\) into the frame \(f_3\) as shown in
The schedulability of packed frames.

As a result, we can obtain two new frames denoted

f2, s10 \}, and finally rename them to

f1 = \{s1, s2, s6, s4, s7\}, f2 = \{s3, s5, s8, s10, s9\} as the mutation results.

3.6 Feasibility and Schedulability Analysis

After completing the above genetic operations, a new generation consisting of packed frames can be obtained. Since the packed frames will be transmitted on the CAN bus, both feasibility and schedulability of the frames should be checked.

The feasibility checking indicates that the size of any packed frame should comply with the standard size of CAN FD. For example, if there is a packed frame with more than 64 bytes signals, some of the signals should be removed to other frames or a new frame randomly. The checking will repeat until all frames are feasible with respect to size constraints as shown in Fig. 8.

The schedulability checking indicates that the worst case response time (WCRT) of any packed frame should be less than or equal to its deadlines. For schedulability analysis, the first step is to assign the frames priorities in such a way that all packed frames can be schedulable. Then, the WCRT of each frame will be calculated. Note that we assume no offset when transmitting frame on the bus for simplicity of analysis. Finally, the calculated WCRT is compared to the frame’s deadline to determine whether the system is schedulable or not. In case that the obtained frame is unschedulable, we propose a novel merging algorithm to improve the schedulability of packed frames.

The complete flowchart for feasibility and schedulability checking is shown in Fig. 8. Below, detailed information about the analysis and improvement of schedulability will be given.

Before analyzing the schedulability, we should transform the array, which is employed to represent an individual, into the frames denoted by F = \{f1, f2, \ldots, fn\} so that the analysis and other operation can be done afterwards.

**Priority assignment policies:** assigning an appropriate priority to each packed frame is important to obtain a schedulable system. For this purpose, the Audsley’s Optimality Priority Assignment (OPA) algorithm [12] is employed as the optimal priority ordering of frames in this study.

**Schedulability test Sch:** A frame is considered to be schedulable if and only if its WCRT is less than or equal to its deadline. The WCRT of a frame \(f_m\) can be calculated by:

\[
R_m = J_m + w_m + C_m, \tag{5}
\]

where \(R_m\) is the WCRT of \(f_m\), \(J_m\) is the queuing jitter, \(w_m\) is the queuing delay and \(C_m\) is the WCTT of \(f_m\). The \(C_m\) can be calculated by Eq. (1). The queuing delay \(w_m\) is a monotonic non-decreasing function and it can be solved using the recurrence relation given below.

\[
w^{n+1}_m = B^{MAX} + \sum_{k \in hp(m)} \left[ \frac{w^n_m + J_k + \tau_{bit}}{T_k} \right] C_k, \tag{6}
\]

where \(B^{MAX}\) corresponds to the transmission time of the longest possible CAN FD frame (i.e., the frame with 64-byte data), \(hp(m)\) is the set of frames with higher priority than \(f_m\), \(T_k\) is the period of the frame \(f_k\), and \(\tau_{bit}\) is the time taken to transmit a bit on CAN bus.

A proper starting value is \(w^0_m = B^{MAX}\). The recurrence relation iterates until either \(J_m + w^{n+1}_m + C_m > D_m\) in which case the message is unschedulable, or \(w^{n+1}_m = w^n_m\) and \(J_m + w^{n+1}_m + C_m \leq D_m\) in which case the frame \(f_m\) is schedulable, and an upper bound on its response time is \(J_m + w^{n+1}_m + C_m\).

The detailed procedure for calculating the WCRT of a frame can be found in [11].

3.7 Proposed Merging Algorithm

If there is a packed frame that cannot meet its deadlines, we propose a merging algorithm as shown in Fig. 9 to improve the schedulability of the frame. The idea is to reduce the interference time (i.e., the \(w_m\) in Eq. (5)) caused by higher priority frames by merging two higher-priority frames into one frame. For example, there is an unschedulable frame \(f_k\). Suppose that there are two high-priority frames over \(f_k\) denoted by \(f_l\) and \(f_m\). Their transmission time are \(C_l\) and \(C_m\), respectively. Because a frame consists of data bits and arbitration bits, the transmission time of each frame can be divided into two parts, i.e., time related to the data bit transmission denoted by \(C_d\) and time related to the arbitration bit transmission denoted by \(C_a\). Therefore, the following equations can be obtained.
Algorithm 1: Merging algorithm
Input: packed frames (output of GA)
Output: repacked frames and schedulability of individual;
flag=0 succeed in merging; flag=1 fail in merging;
while (there is an unschedulable frame \( f_j \) in an individual)
{
    flag=1;
    while (there are two unchosen higher-priority frames than \( f_j \))
    {
        Select two unchosen higher-priority frames randomly;
        if (the two frames can be merged into one frame \( f_k \))
        {
            Calculate period \( H_k \), deadline \( D_k \), and size \( L_k \) of \( f_k \);
            flag=0;
            break;
        }
    }
    if (flag==1)
    {
    The individual is unschedulable;
    break;
    }
    Priority assignment;
    Analyze the schedulability of the individual;
}

Fig. 9  Frame merging algorithm

\[ C_1 = C_{1d} + C_{1a} \]
\[ C_2 = C_{2d} + C_{2a} \]

If the two frames can be merged into one frame denoted by \( f_3 \), its payload will be the summation of the \( f_1 \) and \( f_2 \). Therefore,
\[ C_3 = C_{3d} + C_{3a} = C_{1d} + C_{2d} + C_{3a}, \text{ and} \]
\[ C_1 + C_2 = C_{1d} + C_{1a} + C_{2d} + C_{2a} \]

According to the CAN FD standard, the arbitration field of each frame has the same length and the same transmission rate, which leads to the following results.
\[ C_{1a} = C_{2a} = C_{3a}, \]
\[ C_3 < C_1 + C_2. \]

Based on the above analysis, we may merge two frames which have higher priority than \( f_k \) into a new frame so that the transmission time of the new frame may be less than the summation of the two original frames. As a result, the frame \( f_k \) may be schedulable after merging two frames due to the reduced interference time. Figure 9 lists the complete merging algorithm. Note that the main objective of the merging algorithm is to improve the schedulability of packed frames but not the minimization of bandwidth utilization. Our merging heuristic follows such a strategy in an iterative loop until all frames are schedulable or no frame can be merged further, which means the packed frames are unschedulable.

4. Experimental Results

4.1 Experimental Objective and Setup

All experiments are classified into three parts. The first experiments in Sect. 4.2 is to evaluate the reduction on bandwidth utilization after signal packing especially for varied bus load. We ignore the schedulability test in this part and focus on the capability to reduce the bandwidth utilization. That is, \( \text{Sch}() \) is set to 0 when calculating fitness in the experiments. The second experiments in Sect. 4.3 is to evaluate the improvement on schedulability after signal packing especially for high bus load. In this part, both schedulability analysis and merging algorithm for improving schedulability are conducted. The third experiments in Sect. 4.4 is to evaluate the efficiency and scalability of GA.

In the experiments, the periods and the size of signals were varied between 100ms and 5000ms, and between 1 byte and 14bytes, respectively, with uniform distribution. The arbitration bit rate and data bit rate of CAN FD are set to 500Kbit/s, and 2Mbit/s, respectively, which are the same with the parameters used in [2]. For the parameters of GA, the number of individuals in each generation \( N_{\text{ind}} \) and the number of generations \( N_{\text{gen}} \) are set to 200 and 800, respectively, and the probability of mutation \( p_m \) is set to 0.005. These parameters of GA are chosen in such a way that the algorithm can achieve a good balance between acceptable performance and reasonable computation time.

4.2 Evaluation of Reduction on Bandwidth Utilization

The experiment in this section is to evaluate the efficiency of GA on bandwidth reduction when comparing with no packing and state-of-the-art method. As mentioned earlier, since the CAN FD frame Selection method CaFeS [2] is the state-of-the-art in the literature, we select it for comparison. The CaFeS method is an optimization heuristic that intends to decrease bandwidth waste of the signal communication. The idea of CaFeS is a dynamic programming algorithm to reduce the bandwidth waste with recursive equations. For fair comparison, our approach and the previous method are compared in terms of consumed bandwidth using the same signal sets as described above.

The experimental results are shown in Fig. 10, where “before packing” (i.e., without packing) means each signal is a frame in itself and the size of the resulting frame is the nearest CAN FD frame size that can accommodate it. The number of the signals is chosen from 20 to 200 with an increment of 20 signals. As can be seen, the GA can achieve significantly less bandwidth utilization than the CaFeS, which suggests that the GA is more effective to find out the global optimal solution than the greedy based heuristic algorithm. Moreover, the results also imply that the GA can deal with the varied periods of signals better than existing method.
In the experiment of Fig. 11, the bandwidth utilization of the initial signal sets is divided into 5 categories varying between 0-20%, 20-40%, 40-60%, 60-80% and 80-100%. For each category, three bars are plotted to show the average bandwidth utilization obtained by GA, CaFeS and without packing, respectively. As can be seen from the Fig. 11, the proposed GA can reduce the bandwidth utilization more than 50% and 20% comparing with “without packing” and the state-of-the-art method, respectively in case of high bus load. The convergence speed of GA is shown in Fig. 12, which indicates that the GA can obtain a stable bandwidth after about 300 generation evolution.

### 4.3 Evaluation of Schedulability of Packed Frames

To validate the effectiveness of the proposed GA and merging algorithm, we compare the percentage of schedulable frames of three methods in the experiments. Three methods are GA without schedulability test, GA with schedulability test and CaFeS.

First, experiments were conducted to illustrate the variation of bandwidth utilization and schedulability of packed results when the number of signals is increased from 1 to 300. As can be seen from the results in Fig. 13, the packed frames become to be unschedulable for CaFeS when the number of signals is more than 250, while the result of GA with schedulability test is still schedulable even the number of signals is 300. Moreover, the GA results in lower bandwidth utilization than CaFeS when the packed frames are schedulable.

Second, intensive experiments were conducted by using 20 randomly generated signal sets in each category as listed in Table 1. The periods of signals are selected from

| No. | Packed frames | Original signals |
|-----|---------------|------------------|
| 1   | $0 < bu <= 0.2$ | $0 < bu <= 0.4$  |
| 2   | $0.2 < bu <= 0.4$ | $0.4 < bu <= 0.8$ |
| 3   | $0.4 < bu <= 0.6$ | $0.8 < bu <= 1.2$ |
| 4   | $0.6 < bu <= 1.0$ | $1.2 < bu <= 1.6$ |
| 5   | $1.0 < bu <= 1.8$ | $1.6 < bu <= 1.9$ |
10ms to 1000ms, and the majority is from 10ms to 100ms. The experimental results are summarized in Fig. 14, from which we can draw the following observations.

1. The three methods can achieve the same 100% schedulable frames for low bandwidth utilization (i.e., below 0.6 in the experiments), which indicates the packed frames could be 100% schedulable even without conducting schedulability test for low bus load.

2. For bandwidth utilization more than 0.6, some packed frames tend to be unschedulable. The schedulability of packed results can be improved significantly after applying the merging algorithm (i.e., the GA with schedulability test) comparing with the GA without using merging algorithm and CaFeS. As can be seen in Fig. 14, for packed signal sets with 0.6-0.8 bandwidth, only 77.8% signal sets are schedulable in CaFeS while 100% signal sets are schedulable in our GA. Moreover, in case that the packed bandwidth increases further to 0.8-1, while 66.7% packed signal sets are feasible in our GA, almost no signal set can be schedulable in CaFeS.

Given that most signals have strict real-time constraints when transmitting them on the CAN bus in the safety-critical application, the proposed GA combining with merging algorithm is efficient for reducing bandwidth utilization and meeting the real-time constraints, thus it is applicable in practice.

### 4.4 Evaluation of Execution Time and Scalability of GA

All the above experiments were conducted on a windows7 laptop running on a 3.19GHz Pentium(R) Dual-Core CPU processor with 2GB main memory, and all experiments were finished within 10 hours.

To further evaluate the effect of each GA operation in terms of percentage time, we packed 200 signals and recorded the execution time of each GA operations in Table 2. As can be seen, the crossover and schedulability test are most time-consume operations, which consume 46% and 31% of total execution time, respectively.

Beside the above GA operation, the runtime of GA is dominated by the number of $N_{ind}$ and $N_{gen}$. The two parameters are selective, which can provide a mechanism to tradeoff between the execution time and optimality of the solution in practice. Considering the fact that all experiments were conducted on a normal laptop, we believe that our GA is applicable in practice, and it can be speeded up by taking advantage of current multi-core CPUs and GPUs platform.

### 5. Conclusion

In this paper, we proposed a dedicated genetic algorithm for CAN FD frame packing to minimize the bandwidth utilization and meet the real-time constraints. The proposed algorithms can search the global optimal solution by considering the constraints on frame size, different periods of signals, and deadline of each frame. The experimental results revealed that (1) our GA can achieve significantly less bandwidth utilization for low bus load than existing greedy approximate algorithm; (2) the proposed merging algorithm can improve the schedulability of packed frames for high bus load; (3) the proposed GA is applicable in practice with respect to execution time.

### References

[1] C.R. Houck, J. Joines, and M.G. Kay, "A genetic algorithm for function optimization: a Matlab implementation," NCSU-IETR 95.09, 1995.

[2] U.D. Bordoloi and S. Samii, “The Frame Packing Problem for CAN-FD,” 2014 IEEE Real-Time Systems Symposium, pp.284–293, 2014.

[3] K. Deb, et al., “A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II,” Parallel problem solving from nature PPSN VI, Springer Berlin Heidelberg, 2000.

[4] D. Whitley, “An executable model of a simple genetic algorithm,” Foundations of genetic algorithms, vol.2, pp.45–62, 2014.

[5] J. Grefenstette, J. Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms. Psychology Press, 2013.

[6] F. Polzlbauer, I. Bate, and E. Brenner, “Optimized frame packing for embedded systems,” IEEE Embedded Syst. Lett., vol.4, no.3, pp.65–68, 2012.
[7] P. Roy, P. Roy, and A. Chakrabarti, “Modified shuffled frog leaping algorithm with genetic algorithm crossover for solving economic load dispatch problem with valve-point effect,” Applied Soft Computing, vol.13, no.11, pp.4244–4252, 2013.

[8] R. Saket and N. Navet, “Frame packing algorithms for automotive applications,” J. Embedded Computing, 2(1), 2006.

[9] K. Sandstrom, C. Norstrom, and M. Ahlmark, “Frame packing in real-time communication,” In RTCSA, 2000.

[10] F. Hartwich, “CAN with flexible data-rate,” International CAN Conference, 2012.

[11] R.I. Davis, A. Burns, R.J. Briel, and J.J. Lukkien, “Controller area network (CAN) schedulability analysis: Refuted, revisited and re-visited,” Real-Time Systems, vol.35, no.3, pp.239–272, 2007.

[12] N.C. Andsley and Y. Dd, Optimal Priority Assignment And Feasibility Of Static Priority Tasks With Arbitrary Start Times[J]. 2000.

Shan Ding received the B. E. degree in Department of Computer Science and Application in Northeastern University, Shenyang, China, in 1989, and the M.E. degree in Department of Electrical and Computer Engineering from Nagoya Institute of Technology of Japan, in 1999. He received his Ph.D. degree in computer science from Nagoya Institute of Technology in 2003. From Sep. 2004 to Aug. 2006, he was a visiting postdoctoral researcher in the Department of Information Engineering, the Graduate School of Information Science, Nagoya University, where he was financially supported by JSPS Postdoctoral Fellowship for Foreign Researchers. He is currently an Associate Professor in School of Computer Science and Engineering, Northeastern University, P. R. China. His research interests include algorithm design and analysis, real-time scheduling theory, and embedded system design. He is a member of ACM and CCF.

Gang Zeng is an Associate Professor at the Graduate School of Engineering, Nagoya University. He received his Ph.D. degree in Information Science from Chiba University in 2006. From 2006 to 2010, he was a Researcher, and then Assistant Professor at the Center for Embedded Computing Systems (NCES), the Graduate School of Information Science, Nagoya University. His research interests mainly include power-aware computing and real-time embedded system design. He is a member of IEEE and IPSJ.

Ryo Kurachi is a Designated Associate Professor at Center for Embedded Computing Systems, Nagoya University. He graduated from Tokyo University of Science with undergraduate majors in applied electronics. After a few years working at AISIN AW CO., LTD. as a software engineer, he received his master’s degree from Tokyo University of Science in 2007, followed by his Ph.D. in information science from the Nagoya University in 2012. His research interests includes embedded systems and real-time systems. Within that domain, he has investigated topics such as in-vehicle networks and real-time scheduling theory and embedded systems security. He is a member of IPSJ and IEICE.

Ruifeng Huang is a master course student in School of Computer Science and Engineering, Northeastern University. He received the B.S. degrees in electronics and electrical engineering from Dalian University of Technology in 2014. His research interest is real time scheduling theory.