TDCA: Improved Optimization Algorithm with Degree Distribution and Communication Traffic for the Deployment of Software Components Base on AUTOSAR Architecture

Kunpeng Zhang
Jilin University
Yanheng Liu
Jilin University
Jindong Zhang (zhangjindong_100@163.com)
Jilin University
Guanhua Zhang
Jilin University
Jingyi Jin
Jilin University
Yunhao Li
Jilin University
Fengmin Tang
China Automotive Technology and Research Center

Research Article

Keywords: AUTOSAR, Busload, Components deploying, ECU equalization

Posted Date: December 6th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-1131775/v1

Noname manuscript No.
(will be inserted by the editor)

TDCA: Improved optimization algorithm with degree distribution and communication traffic for the deployment of software components base on AUTOSAR architecture

Kunpeng Zhang[1] · Yanheng Liu[1,2] · Jindong Zhang[1,2,3] · Guanhua Zhang[1] · Jingyi Jin[1] · Yunhao Li[1] · Fengmin Tang[4,5]

Received: date / Accepted: date

Abstract AUTOSAR (Automotive Open System Architecture), as an open, standardized framework for automotive electronic software development, has gradually become the standard followed by major automotive manufacturers and automotive electronic device suppliers. The electronic software system problem improves the development efficiency and portability of the system by reducing the development cost of automotive electronic software while ensuring the quality of products and services, which is beneficial for subsequent upgrades and updates of the system. In order to improve the reliability of the software component deployment algorithm based on AUTOSAR architecture, we proposed the TDCA algorithm. During the execution of the algorithm, communication volume and communication degree are introduced to improve the accuracy of the deployment plan by optimizing the bus load and ECU balancing. Algorithm comparison experiments show that comparing heuristic and linear optimization algorithms, the TDCA algorithm proposed in this paper has significant advantages in reducing bus load and ECU utilization. The algorithm can reduce the communication between cores and balance ECU load according to the constraints of AUTOSAR architecture.

Keywords AUTOSAR · Busload · Components deploying · ECU equalization

Kunpeng Zhang
zkp0113@sina.com
Yanheng Liu
yhliu@jlu.edu.cn
✉ Jindong Zhang
zhangjindong_100@163.com
Guanhua Zhang
344276932@qq.com
Jingyi Jin
jyjin20@mails.jlu.edu.cn
Yunhao Li
lyh2833495@126.com
Fengmin Tang
tangfengmin@catarc.ac.cn

[1] College of Computer Science and Technology, Jilin University, Changchun 130012, China
[2] Key laboratory of symbol computation and knowledge engineering of the ministry of education, Jilin University, Changchun, China
[3] State key laboratory of automobile simulation and control,Jilin University, Changchun, China
[4] College of Mechanical Engineering, Hebei University of Technology, Tianjin 300132, China
[5] China Automotive Technology and Research Center, Tianjin 300300, China

1 Introduction

The traditional development model of automotive electronic software is no longer suitable for today's increasingly complex software systems. There is an urgent need for new and efficient automotive electronic software architecture to replace the traditional automotive electronics software development. In response to the intricate design of automotive electronic systems, leading automotive original equipment manufacturers and Tier 1 suppliers jointly established AUTOSAR, which defines a set of support for function-driven, distributed automotive electronic software development methods and software on ECUs Architecture standards to apply to different automotive ECU platforms [1–3]. AUTOSAR is an open and standardized framework for developing automotive electronic software, with the concept of "cooperating on standards and competing in implementation." The purpose is to solve the renewal and replacement of automotive electronic systems and efficiently carry out more complex tasks. The development of the automotive electronic software system improves the software's reusability, reduces the development cost, and solves low software portability and software integration difficulties [3].

AUTOSAR originated in Europe. At first, it was only applied to some sizeable international automobile manufacturers. The applied products were also minimal, only partially applied in engine control modules, body control modules, and gateway modules [4]. Besides, there were only controllers with relatively simple functions in the car at that time, and the software accounted for a small proportion of the entire car electronic control system [5]. It was easy to transplant a complete software system to different hardware platforms.

The application rate of AUTOSAR is relatively low. However, with the rapid development of new technologies in recent years, more and more artificial intelligence and Internet of Things technologies have gradually begun to be applied to automotive electronic systems, such as automatic parking, remote upgrades, fatigue monitoring, workshop communication, lane monitoring and alarm, Pedestrian detection and other functions. The most significant feature of these technologies is that the software algorithms are very complicated, and the software has become the core of the entire automotive system. Therefore, complex car embedded systems need to integrate various essential or critical application functions, and car systems have been upgraded to distributors with many ECUs(Electronic Control Units) [6]. The number of software components implemented as application functions increases to more than tens of thousands with the increase of automobiles' complex functions. At the same time, it will be more complex while the software executes on the ECU. AUTOSAR defines an architecture based on component standardized interfaces, which improves the reusability of components by extracting functions from the hardware [7].

AUTOSAR contains three layers of the automotive embedded system from top to bottom, shown as Fig.1. Application software layer which composed of software components of different granularities. RTE(Runtime Environment) is the heart of AUTOSAR ECU architecture. It is the realization (for particular ECU) of the interfaces of the AUTOSAR VFB(Virtual Function Bus) RTE maps all the runnable of each component in local ECU to the OS tasks and builds the existing intra-ECU communications among them. The AUTOSAR Basic Software layer(BSW) is mainly used to provide essential software services, including standardized system functions and functional interfaces. It comprises of a series of essential service software components, including system services, memory services, communication services, etc. [4,8]. The deployment of SWC to ECU is an essential step in system configurations [9]. For a system composed of ECUs, there are many feasible solutions for deploying SWC to ECUs. The conventional software component deployment problem has been proved to be NP-hard [10,11]. Therefore, we purpose an improved algorithm to configure and deploy the SWC to ECU instead of manual operation [12]. However, the initial ECU of the CAT [6] algorithm is randomly selected. Facing the SWC with a low communication degree in the complex communication network, because the CAT algorithm cannot effectively optimize the ECU's internal communication traffic, it will increase bus traffic. On the one hand, the initial ECU communication degree is small, and the busload increases, and on the other hand, it will increase the algorithm execution time.

In this paper, we optimized based on the CAT algorithm [6] and proposed a faster and more effective deployment of SWC based on the traffic degree and communication traffic clustering algorithm named TDCA. TDCA has the following unique functions. First, TDCA sorts SWCs by degree distribution statistics, selects k SWCs with the enormous degree value as the initial ECU cluster center, and deploys other SWCs to ECUs. Second, we defined a penalty function in the traffic-dependent clustering process to adjust the ECU load factor's weight value dynamically. We use the current ECU utilization and the parameters defined in advance to calculate the penalty function value's function value. When deploying all SWCs, the system's ECU can balance the load and effectively control the communication network's bus speed. The main contributions of this paper are summarized as follows.

(1)We formulate the optimization deployment problem of SWC to ECU in AUTOSAR architecture to optimize the bus traffic and ECU load.

(2)A clustering optimization algorithm based on degree distribution and traffic (TDCA) is proposed for solving the optimization problem. TDCA introduces the degree value and traffic of SWC to optimize the algorithm performance and algorithm execution time during the deployment of SWC to ECU. The TDCA algorithm has more advantages in optimizing bus traffic and ECU load.

(3)The performance of the proposed method is verified by performing the simulation. For comparison, TDSCA(Traffic-Degree-Subset-Cluster-Algorithm) , a heuristic optimization algorithm and linear optimization algorithm are introduced to evaluate the proposed TDCA algorithm's performance in the deployment of SWC to ECU. Moreover, the stability of the algorithm under different data models was tested.

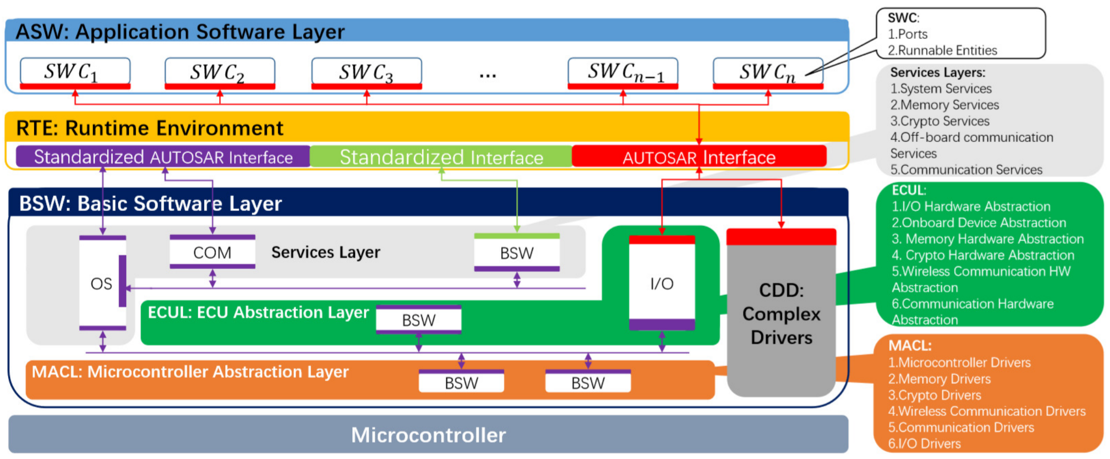
ASW: Application Software Layer
SWC_1
SWC_2
SWC_3
...
SWC_{n-1}
SWC_n
SWC: 1.Ports 2.Runnable Entities
Services Layers: 1.System Services 2.Memory Services 3.Crypto Services 4.Off-board communication Services 5.Communication Services
RTE: Runtime Environment
Standardized AUTOSAR Interface
Standardized Interface
AUTOSAR Interface
BSW: Basic Software Layer
OS
COM
Services Layer
BSW
I/O
CDD: Complex Drivers
ECUL: ECU Abstraction Layer
BSW
ECUL: 1.I/O Hardware Abstraction 2.Onboard Device Abstraction 3. Memory Hardware Abstraction 4. Crypto Hardware Abstraction 5.Wireless Communication HW Abstraction 6.Communication Hardware Abstraction
MACL: Microcontroller Abstraction Layer
BSW
BSW
MACL: 1.Microcontroller Drivers 2.Memory Drivers 3.Crypto Drivers 4.Wireless Communication Drivers 5.Communication Drivers 6.I/O Drivers
Microcontroller

Fig. 1 AUTOSAR architecture

The rest of this paper is organized as follows. Related research is described in Section II. The concept of component-based distributed systems is described in Section III. Section IV formulates the deployment problems and parameter simulation of SWC. Section V proposes the TDCA and TDSCA algorithm. Section VI introduces the experimental evaluation and result analysis. Section VII presents a summary of findings and conclusions.

2 Related research

Most researchers investigating multi-processor task deployment have utilized integer programming [13], simulated annealing and tabu search heuristic algorithm [14], particle swarm optimization [15], dynamic programming [16] and ant colony algorithm [17]. Heuristic algorithm [18], the greedy grouping strategy [19] has essential reference significance for component mapping problems [20, 21]. The CTMC model [22] is used to handle distributed systems' availability and uses a rule-based hierarchical genetic algorithm with random scheduling strategies to complete task deployment [6].

However, the algorithm does not consider load balancing [23, 24]. Dougherty focuses on the deployment of distributed real-time embedded systems of software components [25]. The bin packing algorithm has been used in this problem, but the communication between tasks has not been considered [6].

Zhang Ming, and Z.Gu formulated and solved two optimization problems: mapping SWC to ECU, the purpose is to minimize the busload; for a given SWC to ECU mapping, mapping the runnable entities on each ECU To the OS task, and assign a data consistency mechanism to each shared data item to minimize the memory size requirement on each ECU, while ensuring the schedulability of the task set on all ECUs [17].

Faragardi, Hamid Reza introduced a heuristic algorithm to compare the task granularity of all runnable entities whose tasks are hosted on the same core, which improves the flexibility of assigning runnable entities to each core [20]. Faragardi, Hamid Reza, proposed to reduce the overall network communication time and meet the given timing and priority constraints, without considering the load balance of ECU. Saidi, Salah Eddine, et al. proposed an ILP (Integer Linear Programming) equation for AUTOSAR running mapping process on a multi-core architecture, aiming to minimize inter-core communication and balance the processor load [26]. Ran, Zheng, et al. proposed a traffic-based software component deployment algorithm, CAT algorithm, under the premise of optimizing bus communication costs while balancing the ECU load [6].

3 Background

3.1 Software components and Runnable entities

In the AUTOSAR architecture, the automotive electronics application software consists of software components located on the AUTOSAR RTE [27]. The specific behavior of software components depends on the cooperation of runnable entities. Generally, a software component contains one or more runnable entities. In the AUTOSAR architecture, a runnable entity is a piece of program code used to implement a simple algorithm or a specific function.

There are n SWCs communicating with each other in the automotive software system, denoted by $A = \{a_i \mid i = 1, 2, \ldots, n\}$, let $a_i = \{b_{i,j} \mid j = 1, 2, \ldots, m_i\}$, $b_{i,j}$ is the set of runnable entities of the software component a_i, where $m_{i,j}$ is the number of runnable entities. $b_{i,j} = (P(b_{i,j}), e(b_{i,j}), I(b_{i,j}), O(b_{i,j}))$ among them, $P(b_{i,j})$ is the execution period of $b_{i,j}$, $e(b_{i,j})$ is the worst execution time of a runnable entity $b_{i,j}$, $I(b_{i,j})$ is the order of the runnable entity in the software component of $b_{i,j}$,$O(b_{i,j})$ is the activation offset offset value of $b_{i,j}$.

According to AUTOSAR, all runnable entities are activated by RTE-Event. Two runnable entities are defined in AUTOSAR RTE, Category1: No waiting time, execute immediately, offset=0. Category2: contains at least one waiting time, or when the server calls, it needs to wait for response feedback. There are 12 RTE-Events defined in Classic AUTOSAR. Most of the RTE-Events in AUTOSAR is periodic; that is, they will be triggered at intervals. However, periodic events are divided into RTE-Events with a strict instruction period and RTE-Events with a non-strict instruction period. The underlying timing triggers directly trigger RTE-Events with strict instruction periods. In contrast, other periodic RTE-events are generated by periodic runnable entities, that is, a series of periodic runnable entities may generate a series of events during operation. RTE-event. We assume that all runnable entities are triggered periodically, and each runnable entity is equal to the timer period. We defined the period of the runnable entity $b_{i,j}$ as $P(b_{i,j})$ [6].

In AUTOSAR, the runnable entity is responsible for SWC communication, and its code requirements are as simple as possible to meet reuse requirements and response time constraints. Because the function and structure of the runnable entity are relatively simple, usually, the runnable entity's worst execution time is much shorter than its period $e(b_{i,j}) > P(b_{i,j})$ [6].

3.2 Bus architecture

The SWC communication with each SWC by RTE through such as CAN, local interconnect network(LIN), FlexRay, and automotive Ethernet to provide communication to higher layers through a uniform interface [6, 28–30]. A typical hardware topology composed of three parts, with three buses and 9 ECUs, is shown as Fig.2. The ECU0 and ECU1 connect to BUS0, ECU2, and ECU3 connect to BUS1, and ECU4 to ECU6 connect to BUS1. Gateway0 is a gateway that acts as a bridge between BUS0 and BUS1, and Gateway1 is a gateway that acts as a bridge between BUS0 and BUS2.

In the current automotive electronic system, the hardware topology is more complicated, and the number of buses is more extensive than before. For complex hardware topology, the gateway can be used as a node to isolate and analyze the entire hardware topology. In the experiment, we presume that the ECU has the same

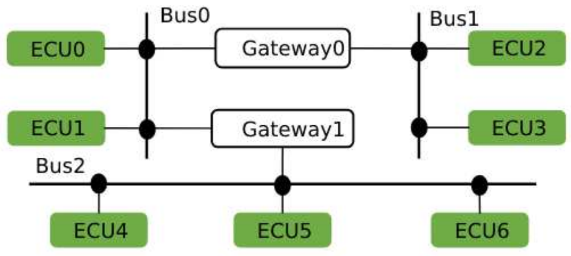

Fig. 2 Example of bus architecture hardware topology

performance indicators on the bus, so we only need to focus on the performance indicators on a single bus in a distributed system.

3.3 Communication Network

AUTOSAR defines different ports and interfaces for communication between SWCs according to the communication direction and communication type, including Provide-Port for data output and Receive-Port data requests. At the same time, AUTOSAR divides the application's overall function into a combination of several SWCs (including atomic SWC and non-atomic SWC). Non-atomic SWC can be divided into multiple interconnected atomic SWC. SWC communicates through runnable located inside SWC. The communication between runnable entities is divided into software component internal communication, and software component communication. Runnable entities deployed in the same software component communicate through shared variables in the software component, and runnable entities deployed in different software components communicate through ports. AUTOSAR software components provide well-defined connection points, namely ports. Three types of ports are defined in AUTOSAR: demand port Require-In, supply port Provides-Out, Provide-Require InOut (introduced in AUTOSAR 4.1) demand ports. Naturally, the AUTOSAR port can refer to the following types of interfaces: send-receive interface Sender-Receiver; client-server interface client-server; mode switch interface mode switch; non-volatile data interface non-volatile data; parameter interface Parameters; Trigger interface trigger. A software component contains multiple interface ports, which are used for data access and function calls.

In this case, the communication network $Com(A, com)$ is defined to represent the communication between SWCs. It is represented by atomic SWC in the communication network. As the data rate between atomic SWC, $Com(A, com)$ is expressed as Com is expressed as $Com = \{com(a_i, a_j) \mid a_i, a_j \in A\}$. Where the definition

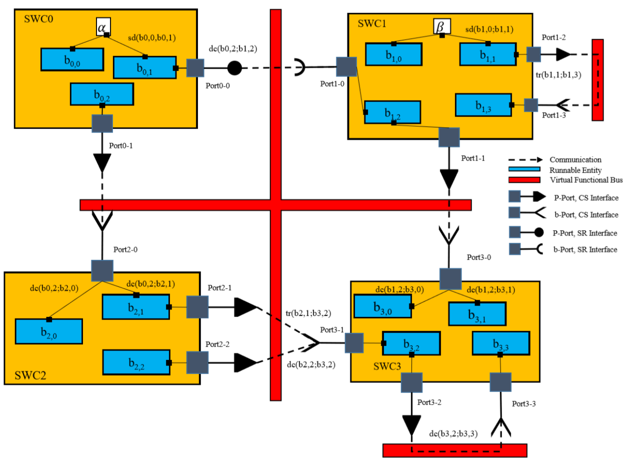

Fig. 3 SWC communications network based on AUTOSAR architecture

$com\left(a_i, a_j\right) = \sum_{p=1,q=1,b_{i,p}\in a_i \wedge b_{j,q}\in a_j}^{p=m_i,q=m_j} \frac{com(b_{i,p}, b_{j,q})}{P_{(b_{i,p})}}$. Define SD, TR, DC as flows.

$$
\begin{aligned}
com\left(b_{i,p}, b_{j,q}\right) &= \sum_{p=1,q=1,b_{i,p}\in a_i \wedge b_{j,q}\in a_j}^{p=m_{i,q}=m_j} \{SD, TR, DC\} \\
&= \sum \begin{cases} \{sd\left(b_{i,p}, b_{j,q}\right)\} \\ \{tr\left(b_{i,p}, b_{j,q}\right)\} \\ \{dc\left(b_{i,p}, b_{j,q}\right)\} \end{cases}
\end{aligned}
$$

Data sharing belongs to SWC internal variables and does not participate in busload. Reduce or optimize the communication relationship between SWC, reduce communication from the SWC trigger. TR: trigger relationship; SD: data sharing; DC: communication relationship. Fig.3 shows the communication network between 4 SWCs.

In Fig.3, the communication traffic between $b_{2,1}$ and $b_{3,2}$ is $tr(b_{2,1}, b_{3,2})$, and the communication traffic between $b_{2,2}$ and $b_{3,2}$ is $dc(b_{2,2}, b_{3,2})$. The communication rate between SWC2 and SWC3 is $\frac{tr(b_{2,1},b_{3,2})}{Period(b_{2,1})} + \frac{dc(b_{2,2},b_{3,2})}{Period(b_{2,2})}$.

4 Problem simulation

The method proposed in this paper is mainly to find the optimal deployment plan for SWC deployment to ECU. Assume that there are n SWCs and need to be deployed to k ECUs in a unique bus architecture. The deployment problem could be transformed into a communication network N cut and optimized into n ($N = (n_1 \cup n_2 \cup \cdots \cup n_n)$) small communication networks and deploy the n communication into k ECUs. In order to obtain better system performance, we need to consider the factors as follows.

4.1 Traffic and traffic degree

There two types of traffic are in AUTOSAR. One is internal traffic in the same ECU, and the other one is bus traffic. The internal traffic means that the SWCs communicate by internal or shared data. The total communication traffic T_{all} is a fixed network that is constant. The transmission of a large amount of data through the bus will lead the bus congestion, and the sharing of data within the ECU will not increase the bus traffic. To reduce the bus congestion caused by data communication, therefore, when we optimize bus traffic in TDCA, we need to make the internal traffic of each $T(ecu_l)$ as more significant as possible. Therefore, the bus traffic T_{bus} as the total traffic minus the traffic inside the ECU $T(ecu_l)$.

$$
T_{bus} = T_{all} - \sum_{l=1}^{k} T\left(ecu_l\right) \tag{1}
$$

4.2 The number of ECU

For selecting the number of ECUs, for the mapping process from SWC to ECU in the AUTOSAR architecture, we define the minimum k value.

$$
k_{min} = \left\lceil \frac{\sum_{i=1}^{n} \sum_{j=1,b_{i,j}\in a_i}^{j=m_i} \frac{e(b_{i,j})}{p(b_{i,j})}}{0.69} \right\rceil \tag{2}
$$

In fact, in the number of deployed ECUs, the minimum k value will lead to the optimal ECU load, which will result in some SWCs without target ECU deployment. Therefore, the value of k is used as a reference value for the minimum number of ECUs, and the number of ECUs is usually more significant than the value of k.

4.3 ECU equalization

Because if the SWC is deployed only based on the amount of communication, it will not only lead to unbalanced ECU utilization among ECUs but also cause specific tasks in the ECUs to fail to be scheduled for execution. Therefore, in the deployment process, we need to use $W(ecu_l)$ the total utilization of the ECU as a reference for the tasks in each ECU that can be executed and scheduled. The $\frac{e(b_{i,j})}{P(b_{i,j})}$ is the utilization of $b_{i,j}$ deployed on the ECU.

$$W\left(ecu_l\right)=\sum_{i=1,a_i\in A\wedge M(a_i)=ecu_l}^{i=n}\sum_{j=1,b_{i,j}\in a_i}^{j=m_i}\frac{e\left(b_{i,j}\right)}{P\left(b_{i,j}\right)} \tag{3}$$

For a distributed operating system, each ECU obtains approximately the same ECU utilization rate to obtain better performance. Now we use the standard deviation of W (Standard Deviation) as a measure of ECU equalization [31].

$$STD(W)=\sqrt{\frac{1}{k}\sum_{l=1}^{k}[E(W)-E(ecu_l)]^2} \tag{4}$$

Where $E(W)$ is the mathematical expectation of $E(ecu_l)$, we treated the STD as the reference for ECU equalization.

5 TDCA algorithm and TDSCA algorithm

In this section, a faster and more effective Traffic-Degree-Clustering-Algorithm and Traffic-Degree-Subset-Clustering-Algorithm are used to deploy SWCs to ECUs. The TDCA and TDSCA algorithm is described as follows.

For evaluating deployment quality for TDCA, we use the utility function $utility\left(ecu_i,a_j\right)$ and to dynamically adjust the ECU load balance and bus traffic. The value of the utility function is directly proportional to the deployment quality of ECU load and traffic bus. $utility(ecu_i,a_j)$ is the product of the communication traffic $ct\left(ecu_i,a_j\right)$ between SWC a_j and ecu_i and fitness value $f(ecu_i)$.

In order to better evaluate TDCA algorithm, we add communication subsets to TDCA algorithm to construct TDSCA algorithm, and optimize the deployment of the relationship between communication subsets of SWCs with large degree value.

Here, the optimization problem equation is as follows. Find all SWC mapping functions $M(a_i)=ecu_l$.

Goal 1: Minimize bus communication traffic in the system, that is find the maximum internal traffic of all ECUs, $\sum_{l=1}^{k}T\left(ecu_l\right)$.

Goal 2: On the premise of ensuring that each ECU can be dispatched, ECU load balancing is realized.

5.1 T-Traffic

Define $ct(ecu_i,a_j)$ as the traffic between ECU ecu_i and SWC a_j, which represents the communication traffic

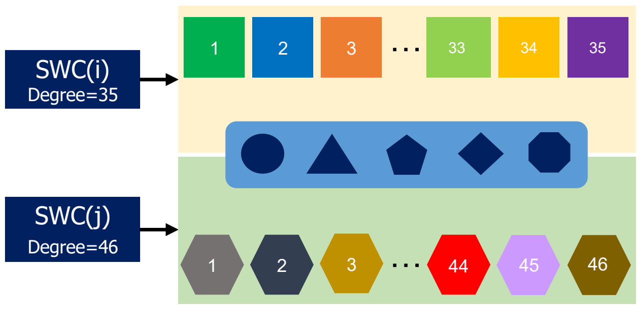

Fig. 4 Communication subset Description.

between current software component a_j and the all SWCs that deployed on ECU ecu_i.

$$ct\left(ecu_i,a_j\right)=\sum_{\forall a_q\in A,M(a_q)\in ecu_i}com\left(a_q,a_j\right) \tag{5}$$

5.2 D-Degree

In the TDCA process deployment, we defined the traffic degree D_i the value that SWC has a communication relationship with other SWCs of each SWC.

$$D_i=\begin{cases}\sum 1, & \text{if} \quad Net(a_i,a_j)\neq 0,i,j\in(1,n);\\ 0, & \text{if} \quad Net(a_i,a_j)=0,i,j\in(1,n);\end{cases} \tag{6}$$

TD is defined as the degree matrix of the network. The larger the TD value, the more other SWCs are communicating with the SWC. Therefore, selecting a larger SWC can ensure that the SWC that has a close communication relationship with the SWC is deployed to the same ECU, which can effectively increase the internal communication traffic of the ECU, reduce the busload and algorithm execute time.

$$TD=\begin{bmatrix}D_1 & \dots & 0\\ 0 & \dots & 0\\ 0 & \dots & D_n\end{bmatrix}=\begin{bmatrix}\sum_{j=1}^{j=n}d_{1,j} & \dots & 0\\ 0 & \dots & 0\\ 0 & \dots & \sum_{j=1}^{j=n}d_{n,j}\end{bmatrix} \tag{7}$$

5.3 S-Subset

The communication relationship in modern automotive electronics is complex, and the same communication targets may exist between SWCs with multiple communication objects. Therefore, we introduced the concept of communication subsets and defined communication subsets based on the degree of communication, as shown in Fig.4.It means that the different SWCs with large degree may have some common communication relationship with others. For example there are 35 degrees and 46 degrees for a_i and a_j, we can inform that from Fig.4 that the communication subset of a_i and a_j is $Subset(a_i, a_j) = 5$, and it shown in the middle part colored with dark blue in Fig.4. The definition of subset is shown in formula(8).

$$\mathrm{Subset_{i,j}} = \sum_{i,j\in(1,n);i\neq j} s_{i,j} = \sum_{i,j\in(1,n);i\neq j} a_{i,(1:n)} .* a_{j,(1:n)} \tag{8}$$

5.4 C-Cluster algorithm

The fitness function $f(ecu_i)$ can automatically adjust the effectiveness function value in the clustering process. At the beginning of the algorithm, the deployment of SWC is mainly based on ECU internal communication. As ECU utilization increases, the fitness function value decrease rapidly. Therefore, the deployment of SWC gradually depends on the ECU load as the algorithm executes. In the end, both bus traffic and ECU load have been optimized.

In the iterative calculation process of the TDCA algorithm, the initial ECUs will select by degree value. The SWC will be deployed to the ECU with the maximum utility value if $Judge < 0.69$. After completing the iteration, update the initial SWC and communication traffic in the ECU, respectively, and redeploy until no better deployment plan can be obtained. At the beginning of the TDCA algorithm, the k SWCs with an enormous degree value are selected as the cluster center points of the initial k ECUs.

The specific operation is as follows, select $\{C_1, C_2, ..., C_k\}$ to become the current cluster corresponding to the ECU. For SWC a_i that has not been deployed to any ECU, recalculate the distance from a_i to all clusters to obtain $\{ct(ecu_1, a_i), ct(ecu_2, a_i), \ldots, ct(ecu_k, a_i)\}$. Then calculate the utility value of each SWC through the fitness function, $\{utility(ecu_1, a_i), \ldots, utility(ecu_k, a_i)\}$. SWC a_i will be deployed on the ECU with the highest utility value.

According to the RM schedule strategy in 1973 [32], the ECU will be non-schedulable when the utilization is more significant than 0.69. Therefore, the utilization of each ECU and the ECU's average utilization should both be less than 0.69. Assuming that SWC a_i is deployed on ECU ecu_j, SWC a_i included in cluster C_j, cluster C_j will become more extensive, and then the distance will be updated. $C_j = C_j \cup a_i$, and the utilization rate of ECU ecu_j has been updated. The $Judge$ is a function to verify whether the ECU task can be scheduled when the SWC is deployed to the ECU.

$$Judge = W(ecu_j) + \sum_{\forall b_{i,q}\in a_i} \frac{e(b_{i,q})}{P(b_{i,q})} \tag{9}$$

$$W(ecu_j) = \begin{cases} Judge, & \text{if} \quad Judge < 0.69; \\ W(ecu_j), & \text{else;} \end{cases} \tag{10}$$

The bus traffic T_{bus} needs to be calculated to determine whether the stop condition is met after the iteration. If necessary, to continue the iterative calculation, the SWC with the enormous internal communication traffic in each ECU will become the new cluster center for the next iteration.

5.5 Fitness function

To ensure that each ECU is schedulable and ECU equalization in the TDCA algorithm, the $f(ecu_i)$ is defined to correct the utility value in the TDCA algorithm, shown as follows.

$$\begin{aligned} f(ecu_i) &= 1 - \left(\frac{W(ecu_i)}{EW + \eta}\right)^3 \\ &= 1 - \left(\frac{k * W(ecu_i)}{\eta + \sum_{i=1}^{n} \eta_{j=1,,i,j\in c_i}^{j=m_i} \frac{e(b_{i,j})}{p(b_{i,j})}}\right)^3 \\ & EW \in (0, 0.69], \eta \in [0, 0.69 - EW] \end{aligned} \tag{11}$$

Among them, in order to set the weight of the TDCA algorithm execution process bus traffic and ECU load, the artificial setting adjustment parameter η is defined. The more significant value of η means the internal communication traffic gets more weight; a smaller value means the ECU load gets more weight. The internal communication traffic of the ECU and inversely proportional to the ECU load.

$f(ecu_i)$ is a monotonically decreasing concave function, mainly used to control the ECU load to ensure that it can be scheduled. When the value of $W(ecu_i)$ is less than $(EW - \eta)$, because $f(ecu_i)$ decreases slowly, $f(ecu_i)$ has little effect on the effectiveness function. At this time, the TDCA algorithm is mainly based on the amount of communication. When $W(ecu_i) \in [EW - \eta, EW + \eta]$, $f(ecu_i)$ decreases rapidly. Due to the rapid decrease of $f(ecu_i)$, SWC deployment will be different from ECU. ECU load becomes the main factor of cluster quality. When $W(ecu_i)$ is greater than $EW + \eta$, the function $f(ecu_i)$ is less than zero, and the fitness function is negative. This means that the load of ECU ecu_i is very heavy, and no SWC will be deployed on this ECU. Therefore, the adaptive function can ensure the schedulability of ECUs and achieve a good balance in each ECU.

5.6 Penalty function

In addition to defining a fitness function similar to CAT, TDCA also defines a penalty function to correct the bus traffic weight for each ECU. In order to ensure the ECU load balance and effectively reduce the communication between ECUs, especially the penalty function item is added on the basis of the fitness function. The penalty function is used to dynamically adjust the ECU load while affecting the communication between ECUs. The penalty function is shown as follows.

$$F(ecu_i) = f(ecu_i) + \alpha_1 * max[0, W(ecu_i) - EW] + \alpha_2 * min[0, (\frac{\frac{T_{all}}{k} - T_{ecu_i}}{T_{bus}})] \quad (12)$$

In formula(12),α_1 is the penalty factor for utilization, and α_2 is the penalty factor for bus load. To be fair, we respectively defined $\alpha_1 = \alpha_2 = 0.5$; in this way, the weights assigned to ECU utilization and bus load remain constant.

5.7 Algorithm analysis

TDCA algorithms describe as follow steps.

```
Step1: Initialized parameters
  $Net[n][n]$;$e[b_{i,j}]$;$P[b_{i,j}]$;$MNI$
  go to Step2.
Step2: Calculate Each SWCs degree and update TD
  $D_i = \sum_{i=1,j=1 \forall Network(a_i,a_j)>0}^{i=n,j=n} 1$;
  Sort TD by descending;
  Select $k$ SWCs from large to small in TD as the initial ECU centers;
  go to Step3.
Step3:Deploy the SWC to ECU
  for each SWC $a_i$, not a medoid
    calculate $utility[ecu_j] = ct[ecu_j][a_i] * f(u_j)$
  sort $utilities[k]$
  if $Judge < 0.69$
    $ecu_{max} = ecu_j$;
    deploy $a_i$ to $ecu_{max}$;
    update $M[a_i] = ecu_{max}$;
    $W[ecu_i] = [ecu_{max}] + \sum_{\forall b_{i,q} \in e_i} \frac{e[b_{i,q}]}{P[b_{i,q}]}$
    $ct[ecu_{max}][a_j] = ct[ecu_{max}][a_j] + Net[a_i][a_j]$;
  go to Step4.
  else
    $ecu_{max} = ecu_{max-1}$;
  go to Step3.
Step4:Calculate the bus traffic $T_{bus}$
  find the maximum distance $ct[ecu_i][a_{max}]$ from $ct[ecu_i][a_1]$ to $ct[ecu_i][a_n]$;
  let $a_{max}$ be medoid of $ecu_i$ and map $a_{max}$ to $ecu_i$, $M[a_{max}] = ecu_i$;
  update $W[ecu_i] = \sum_{\forall b_{\max,q} \in A_{\max}} \frac{e[b_{\max,q}]}{P[b_{\max,q}]}$;
  calculate the value of $\frac{F}{f}$ from $ecu_i$ to $ecu_k$.
  if $\frac{F}{f} > 1$
    $f = \frac{F}{f}$
  else
    $f = f$
  update the new centers of each ECU and set the initial $ct[ecu_{max}][a_j] = ct[ecu_{max}][a_j]$;
Step5:
  if exit $(T_{bus}) \leq MNI$
    end the algorithm.
  else
    go to Step3.
```

Table 1 Parameters definition table

Abbreviation	Full form
$Net[n][n]$	Communication network of SWCs
a_i	Software Components
$b_{i,j}$	Runable Entity
$e(b_{i,j})$	Wrost execution time of $b_{i,j}$
$P(b_{i,j})$	Period of $b_{i,j}$
MNI	Max number of Iteration
D_i	Degree of SWC
TD	Degree matrix of Network
$Subset_{i,j}$	Communication subset
F	Penalty function
ecu	Electronic Control Unit
$ct[ecu_j][a_i]$	Communication traffic between ecu_j and a_i
$W(ecu_i)$	Utilization of ECU
M	Deployment plan of algorithm

$CT[k][n]$ is a temporary two-dimensional array used to store the communication traffic of a_i and SWCs that have been deployed to the current ECU ecu_j, where $CT[ecu_j][a_i] = ct(ecu_i, a_j)$. $W[k]$ is a temporary array of length k used to store the utilization of each ECU. $utility[k]$ is an array of length k used to store the validity function value, where $utility[ecu_i] = u * f(ecu_i, *)$. The array $M[n]$ is used to store the mapping result.

The $exit(T_{bus})$ records the bus traffic T_{bus} during each iteration to determine whether to end the iteration. The TDCA algorithm consists of 5 Steps: In Step1, initialize all variables. In Step2, TD will be calculated. In Step3, deploy SWCs to ECUs. In Step4, the W and ct will be updated. In Step5, if there is no smaller T_{bus}, or the iteration period is reached, the algorithm ends.

In order to verify whether a SWC with a larger degree and a communication subset needs to be de-

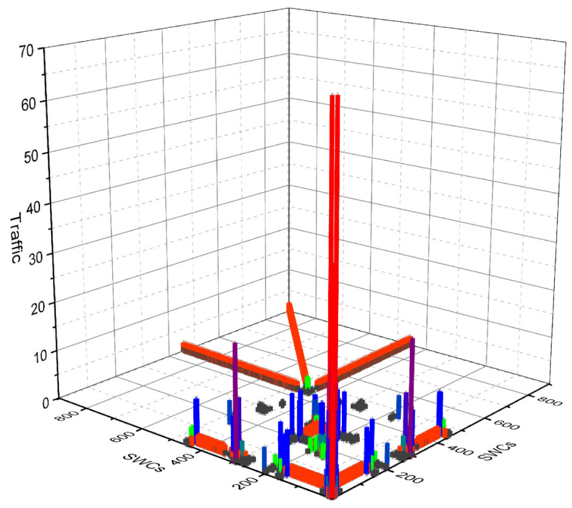

Fig. 5 AIS data set and communication network

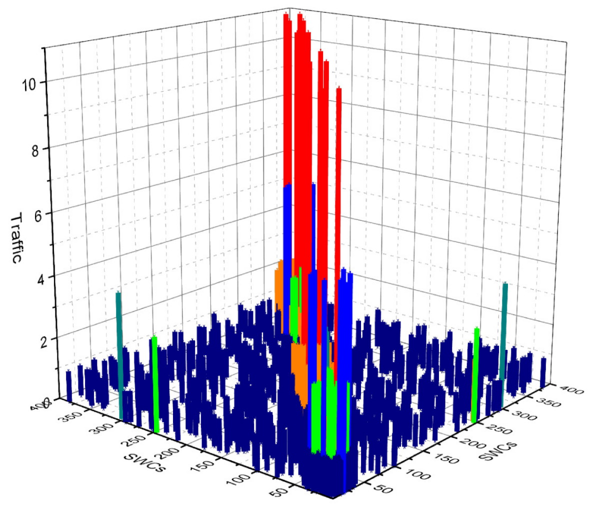

Fig. 6 PLP data set and communication network

ployed to the same ECU, we added the concept of communication subset on the basis of TDCA, constructed the TDSCA algorithm, and calculated the degree value and communication subset for each SWC. During the experiment, the SWC with more communication subsets was mapped to an ECU to obtain experimental data based on the TDSCA algorithm, and shown in Fig8,9,10,11,12,13.

6 Performance evaluation

This section conducts an experimental evaluation of the TDCA, TDSCA, CAT, and heuristic SPA algorithms and compares the results with ILP-based load balancing and ILP-based non-load balancing algorithms.

6.1 Dataset description

The data set AIS is extracted from the application program interface example and has five complex functions provided by the AUTOSAR specification. Based on the communication network of the data set AIS, we generated 904 atomic SWCs. The statistical analysis of the data set AIS shows that AIS's communication density is $\lambda = 0.14$. The data set AIS is shown in Fig.5.

For the artificially synthesized automobile electronic software component set, firstly generate a random connection diagram according to the number of components, and then transform it into a component communication network. Since components are mainly used for communication, runnable entities with enormous communication traffic have longer execution times, and vice versa. Therefore, the executable entities' execution time and period are combined according to the components communication traffic. When generating the executable entities' execution time and period in the component, the average utilization rate of all ECUs must be less than 0.69. In actual automotive electronic applications, the average utilization rate of ECUs will not be too high.

For the synthetic experimental data set, we convert the random connected graph into a specified number of SWC communication network and generate the worst execution time and period of SWC according to the traffic. The worst execution time of runnable entities in SWC is proportional to SWC's communication traffic.

In the experiment process, the PLP model (Planted l-partition model) planned by the graph segmentation algorithm plan using the implantable segmentation model is a multi-level Erdos Renyi model [33] to generate random connected graphs and is expressed as (n, k, p, q). The data set we generated based on the PLP model is shown as Fig.6.

In the PLP mode, the degree of the vertex approximately obeys a random distribution. Because the software fitting distribution in the AUTOSAR architecture obeys the exponential distribution, we generate another set of experimental data (RGE: random graph follows the exponential distribution [33]), the parameters are set to n SWC, the communication network distribution $\lambda = 0.14$. Fig.7 shows an example of the parameter RGE (400,0.14).

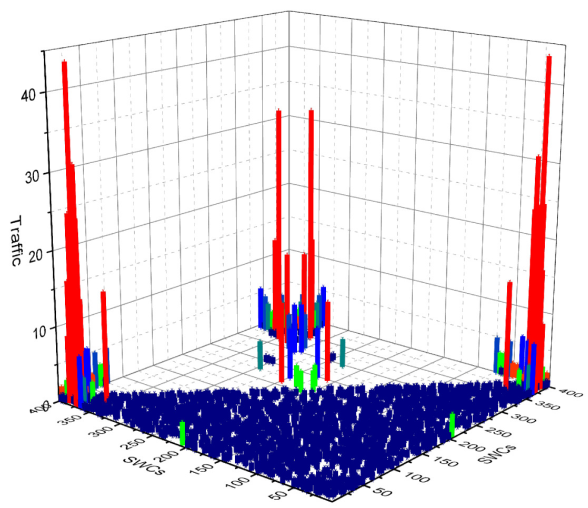

Fig. 7 RGE data set and communication network

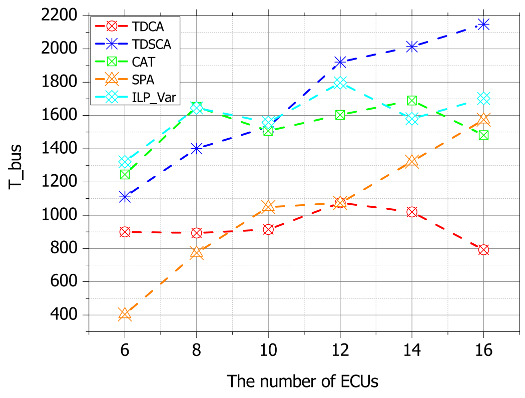

Fig. 8 Bus traffic of AIS with algorithms

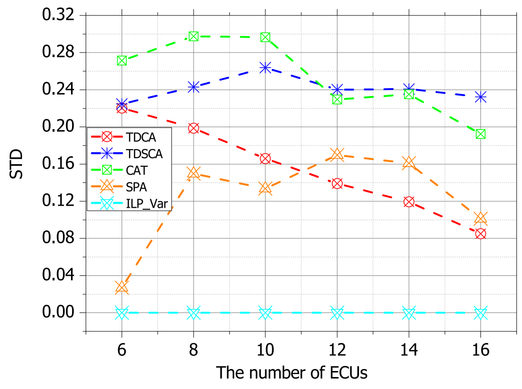

Fig. 9 The Standard Deviation of ECU utilization with algorithms

6.2 Comparison of algorithms with ECU equalization

For the data set AIS, four algorithms are used to deploy SWCs to 6,8,10,12,14 and 16 ECUs. For CAT, SPA, TDSCA, and TDCA algorithms, η is set to $EW/2k$. For ILP_Var, we set the max utilization of each ECU equal to the overall utilization average.

Evaluate the performance of algorithms of deployment base on bus load(T_{bus}), ECU balancing(STD), and the algorithm execution time(Runtime). Fig.8 shows the effect of the number of ECUs on bus traffic. TDCA's bus traffic from 700 to 1100, the value of TDSCA ranges form 1100 to 2200, the bus traffic range of CAT is between 1200 and 1700, the value of SPA ranges from 400 to 1600, and the value of ILP_Var ranges from 1300 to 1800. Compared with other algorithms, the optimization performance of TDCA algorithm for bus traffic is better than other algorithms, and the fluctuation is smaller.

ECU load balancing is shown in Fig.9. Since the maximum utilization of each ECU is manually set in the initial stage, the ILP_Var algorithm has significant performance in ECU load. However, we can inform the ILP_Var could not decrease the bus traffic T_{bus} from Fig.9. For the AIS data set, as the number of ECUs increases, in addition to ILP_Var, the best performance in ECU equalization is the TDCA algorithm, followed by the SPA algorithm, TDSCA algorithm and finally, the CAT algorithm.

The execution time of algorithms is shown in Fig.10. Compared with CAT, TDSCA, SPA, and ILP_Var, TDSCA has the longest execution time, the execution time of the SPA algorithm is in the middle, while TDCA, CAT, and ILP_Var are all less than 100ms, and TDCA is between CAT and ILP_Var in most times.

Fig.8, Fig.9, Fig.10 and Table 2 show that the TDCA algorithm has good performance in reducing bus traffic, optimizing ECU equalization, and algorithm execution time during the deployment process based on the AIS data set. Compared with other algorithms, TDCA algorithm is the best in terms of comprehensive bus load, ECU balance and algorithm running time.

For random experimental data sets generated based on the PLP model and the RGE model. Deploy these SWCs to 8 ECUs through CAT, TDCA, TDSCA and SPA algorithms. Analyze deployed algorithms' performance by comparing bus traffic, the standard deviation of ECU utilization, and algorithm execution time. Fig.11, Fig.12, and Fig.13(a) show the results of the PLP-based model, and Fig.11, Fig.12, and Fig.13(b) show the RGE model data results.

1.Bus traffic: For the communication network generated by the PLP model, there are initially eight closely connected clusters. The number of clusters is equal to

Table 2 Performance of each Algorithm for AIS.

Algorithm	T_{bus}						Std						Runtime					
	6	8	10	12	14	16	6	8	10	12	14	16	6	8	10	12	14	16
CAT	4	5	3	3	4	3	5	5	5	4	4	4	3	3	3	3	3	3
SPA	1	2	1	1	2	2	2	2	3	3	3	4	4	4	4	4	4	4
ILP_Var	5	4	5	4	3	4	1	1	1	1	1	1	1	1	1	1	1	2
TDSCA	3	3	4	5	5	5	4	4	4	5	5	5	5	5	5	5	5	5
TDCA	2	1	2	2	1	1	3	3	3	2	2	2	2	2	2	2	2	1

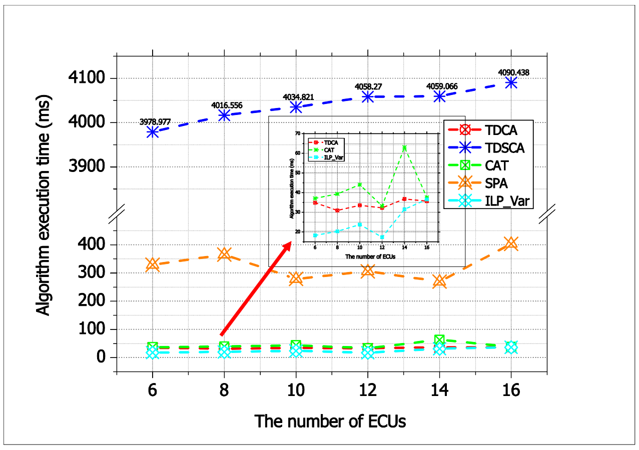

Fig. 10 The execution time of AIS with algorithms

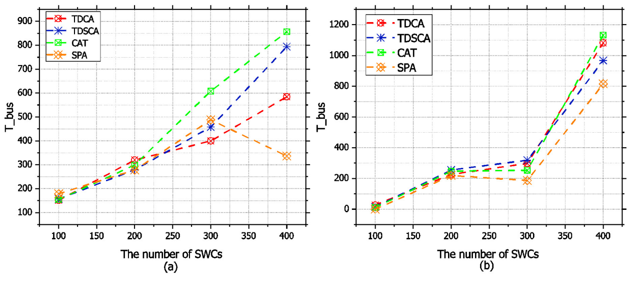

Fig. 11 Bus traffic of PLP and RGE with algorithms

the number of ECUs, so the SPA algorithm obtains the smallest T_{bus} busload. Regardless of the number of SWCs, TDCA will achieve better performance in terms of busload. These two methods effectively aggregated eight original partitions, and the result is shown in Fig.11(a). For the data set based on the RGE model, no matter how many SWCs are deployed, the algorithms will get an acceptable T_{bus} and the result is shown in Fig.11(b).

2.ECU equalization: Fig.12(a), (b) and Table3 show that no matter what the experimental data set is, the standard deviation of TDCA is the smallest due to its fitness function and penalty function. Therefore, TDCA has the best performance in balancing ECU load.

3.Algorithm execution time: Fig.13 shows the execution time, regardless of the communication data under the PLP model or the communication data under

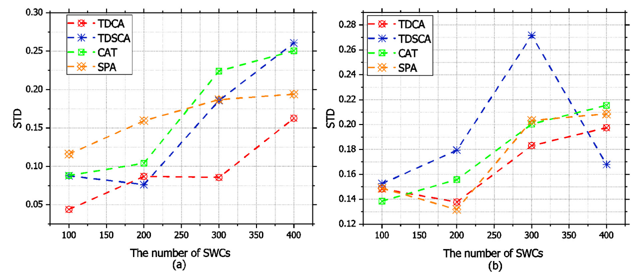

Fig. 12 The Standard Deviation of ECU utilization

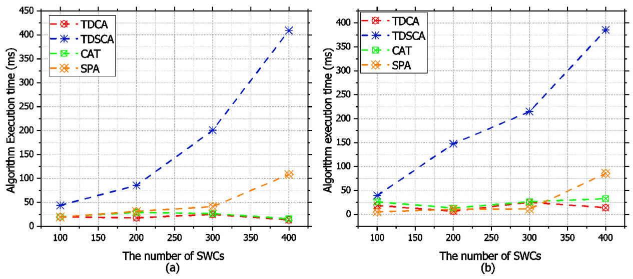

Fig. 13 Execution time of PLP and RGE with algorithms

the RGE model, the CAT algorithm, TDCA algorithm, TDSCA algorithm and SPA algorithm is acceptable when the number of SWCs is small. However, when the number of SWCs increases, the TDSCA and SPA algorithm's execution time increases significantly, especially for TDSCA, while the TDCA algorithm and CAT algorithm still have significant advantages in algorithm execution time. Compared with other algorithms, the TDCA algorithm also has advantages in algorithm execution. In summary, compared with algorithms that meet ECU load balancing, whether it is the AIS data set or randomly generated experimental data set, the TDCA algorithm has advantages in reducing busload, ECU load balancing, and algorithm execution time. With the number of ECUs and SWCs increasing, the TDCA is more stable than others.

6.3 Comparison of algorithm without ECU equalization

Since the clustering algorithm may not be the best, we set up another experiment to compare and analyze the

Table 3 Performance of each Algorithm for PLP and RGE.

Data	SPA			CAT			TDSCA			TDCA		
	T_{bus}	Std	Runtime	T_{bus}	Std	Runtime	T_{bus}	Std	Runtime	T_{bus}	Std	Runtime
PLP100	4	4	1	2	3	3	3	2	4	1	1	2
PLP200	1	4	3	3	3	2	2	1	4	4	2	1
PLP300	3	3	3	4	4	5	2	2	4	1	1	1
PLP400	1	2	3	4	3	5	3	4	4	2	1	1
RGE100	1	3	1	2	1	3	3	4	4	4	2	2
RGE200	1	1	2	3	3	3	4	4	4	2	2	1
RGE300	1	3	1	2	2	3	4	4	4	3	1	2
RGE400	1	3	3	4	4	2	2	1	4	3	2	1

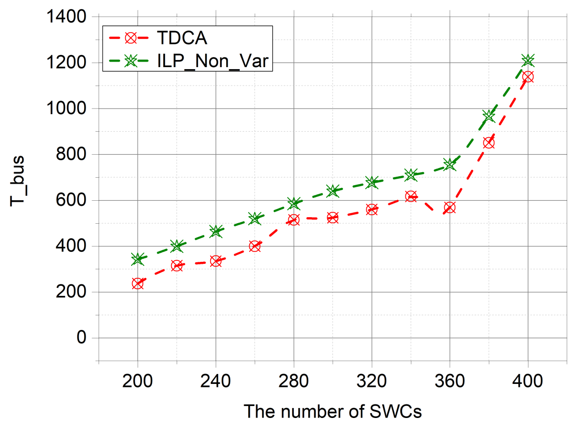

Fig. 14 Bus traffic with TDCA and ILP_Non_Var

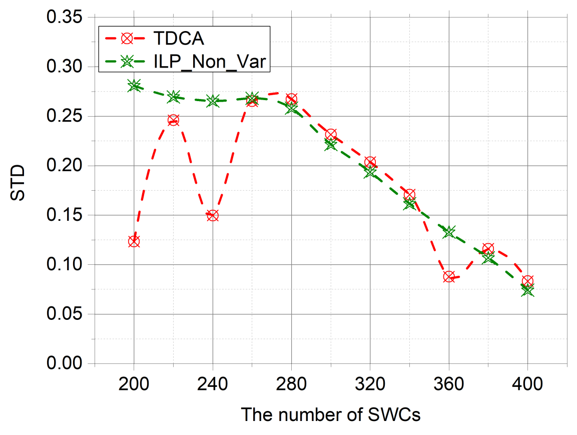

Fig. 15 The Standard Deviation of ECU utilization with TDCA and ILP_Non_Var

TDCA algorithm and the ILP_Non_Var (integer linear programming without ECU equalization). In the deployment process of ILP_Non_Var, we only need to ensure that the utilization rate of each ECU after the algorithm is executed less than 0.69. Compared with the ILP_Var algorithm, the ILP_Non_Var algorithm cannot optimize the ECU load balance, so ILP_Non_Var has poor performance in ECU load optimization shown as Fig.15. This part uses ILP_Non_Var algorithm and TDCA algorithms to deploy the 200 to 400 SWCs that conform to the RGE model distribution, and the number of ECUs is set to 5. Fig.14, Fig.15, and Fig.16 respectively show the bus communication, standard deviation, and algorithm execution time of each algorithm. Fig.14 shows that TDCA is better than ILP_Non_Var in bus traffic. In terms of ECU equalization, the TDCA algorithm's advantages gradually weaken as the number of SWCs increases, as shown in Fig.15. For the algorithm execution time, as the number of SWC increases, the gap between TDCA and ILP_Non_Var algorithm execution time is getting smaller and smaller. Finally, the algorithm execution time of the TDCA algorithm is beyond ILP_Non_Var.

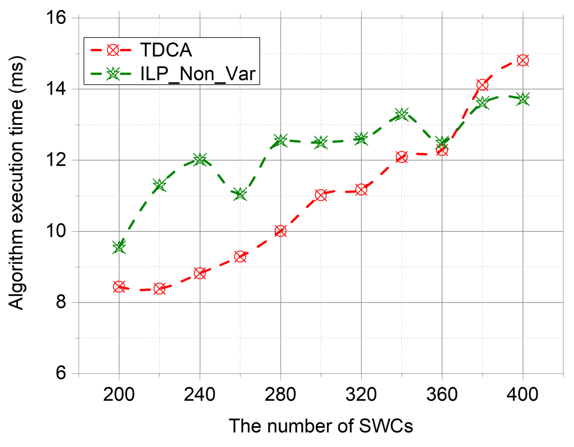

Fig. 16 Algorithm execution time With TDCA and ILP_Non_Var

7 Conclusion

This paper proposes a novel TDCA algorithm is suitable for the SWC deployment algorithm of AUTOSAR architecture. TDCA introduced two improvement factors to improve the busload and ECU load balance of the deployment plan, which software components can be automatically deployed to ECUs in automotive electronic systems. This algorithm can reduce the bus traffic while ensuring schedulability, and get a good performance of ECUs equalization. Compared with CAT, SPA, TDSCA and ILP algorithms, TDCA has an excellent performance in reducing bus traffic and ensuring ECU equalization. The algorithm execution time of TDCA is better than others. Under normal circumstances, the TDCA algorithm can meet the deployment requirements of SWC to ECU under the AUTOSAR architecture.

In actual situations, it is more complicated in the deployment process of SWC to ECUs with a strict end-to-end time limit for deployment. In the paper, the SWC is deployed into the ECU without considering the final time limit of the SWC. In this case, the SWC with a strict end-to-end execution time limit can be constructed into a composite SWC before SWC deployment. In this way, regardless of the communication rate of the two SWCs, the two SWCs will be deployed to identical ECU. An ideal method is to consider the end-to-end execution deadline of the SWC in analyzing the SWC with greater relevance. In the follow-up research work, the objective function will be optimized and adjusted according to this factor to achieve the desired effect.

Acknowledgements This work was funded by the National Natural Science Foundation of China(61872158, 62172186), the National Key Research and Development Program of China (2017-YFB0102503), Jilin Province Science and Technology Development Plan(Key Science and Technology Research)Project(20210201072GX), and Jilin Province Science and Technology Development Plan (International Cooperation) Project(20190701019GH).

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Jorge Martinez, Ignacio Sañudo, and Marko Bertogna. End-to-end latency characterization of task communication models for automotive systems. *Real-Time Systems*, 56(3):315–347, 2020.
2. Anand Bhat, Soheil Samii, and Ragunathan Rajkumar. Practical task allocation for software fault-tolerance and its implementation in embedded automotive systems. *Real-Time Systems*, 55(4):889–924, 2019.
3. Haibo Zeng and Marco Di Natale. Efficient implementation of autosar components with minimal memory usage. In *IEEE International Symposium on Industrial Embedded Systems*, 2012.
4. Ernest Wozniak, Asma Mehiaoui, Chokri Mraidha, Sara Tucci-Piergiovanni, and Sébastien Gerard. An optimization approach for the synthesis of autosar architectures. In *2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA)*, pages 1–10. IEEE, 2013.
5. Wei Peng, Hong Li, Min Yao, and Zheng Sun. Deployment optimization for autosar system configuration. In *International Conference on Computer Engineering & Technology*, 2010.
6. Zheng Ran, Hua Yan, Huimin Zhang, and Yun Li. Approximate optimal autosar software components deploying approach for automotive e/e system. *International Journal of Automotive Technology*, 18(6):1109–1119, 2017.
7. Daehyun Kum, Gwang Min Park, Seonghun Lee, and Wooyoung Jung. Autosar migration from existing automotive software. In *Control, Automation and Systems, 2008. ICCAS 2008. International Conference on*, 2008.
8. Gia Nghia Vo, Richard Lai, and Mohit Garg. Building automotive software component within the autosar environment - a case study. In *International Conference on Quality Software*, 2009.
9. Rajeshwari Hegde, Geetishree Mishra, and K. S. Gurumurthy. An insight into the hardware and software complexity of ecus in vehicles. *Communications in Computer & Information Ence*, 198:99–106, 2011.
10. Guojun Shen, Yanheng Liu, Geng Sun, Tingting Zheng, Xu Zhou, and Aimin Wang. Suppressing sidelobe level of the planar antenna array in wireless power transmission. *IEEE Access*, pages 6958–6970, 2019.
11. Tingting Zheng, Yanheng Liu, Geng Sun, Lin Zhang, Shuang Liang, Aimin Wang, and Xu Zhou. Iwormlf: Improved invasive weed optimization with random mutation and lévy flight for beam pattern optimizations of linear and circular antenna arrays. *IEEE Access*, 8:19460–19478, 2020.
12. Yang Yang. *Software synthesis for distributed embedded systems.* PhD thesis, UC Berkeley, 2012.
13. Ralf Niemann and Peter Marwedel. An algorithm for hardware/software partitioning using mixed integer linear programming. *Design Automation for Embedded Systems*, 2(2):165–193, 1997.
14. Petru Eles, Zebo Peng, Krzysztof Kuchcinski, and Alexa Doboli. System level hardware/software partitioning based on simulated annealing and tabu search. *Journal of Design Automation for Embedded Systems*, 2(1):5–32, 1997.
15. Alakananda Bhattacharya, Amit Konar, Swagatam Das, Crina Grosan, and Ajith Abraham. Hardware software partitioning problem in embedded system design using particle swarm optimization algorithm. In *International Conference on Complex*, 2008.
16. Ji Gang Wu, Thambipillai Srikanthan, and Guang Wei Zou. New model and algorithm for hardware/software partitioning. *Journal of Computer Science & Technology*, 23(4):644–651, 2008.
17. Ming Zhang and Zonghua Gu. Optimization issues in mapping autosar components to distributed multithreaded implementations. In *IEEE International Symposium on Rapid System Prototyping*, 2011.
18. Fabrizio Ferrandi, Pier Luca Lanzi, Christian Pilato, Donatella Sciuto, and Antonino Tumeo. Ant colony heuristic

for mapping and scheduling tasks and communications on heterogeneous embedded systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(6):911–924, 2010.

19. Eduardo HM Cruz, Matthias Diener, Laércio L Pilla, and Philippe OA Navaux. An efficient algorithm for communication-based task mapping. In *2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 207–214. IEEE, 2015.
20. Hamid Reza Faragardi, Bjorn Lisper, and Thomas Nolte. Towards a communication-efficient mapping of autosar runnables on multi-cores. In *Emerging Technologies & Factory Automation*, 2013.
21. Hamid Reza Faragardi, Björn Lisper, Kristian Sandström, and Thomas Nolte. An efficient scheduling of autosar runnables to minimize communication cost in multi-core systems. In *7th International Symposium on Telecommunications (IST)*, 2014.
22. Shampa Chakraverty and Anil Kumar. A rule-based availability-driven cosynthesis scheme. *Design Automation for Embedded Systems*, 11(2-3):193–222, 2007.
23. Yecheng Zhao and Haibo Zeng. The concept of maximal unschedulable deadline assignment for optimization in fixed-priority scheduled real-time systems. *Real-Time Systems*, 55(3):667–707, 2019.
24. Hélène Martorell, Jean-Charles Fabre, Matthieu Roy, and Régis Valentin. Improving adaptiveness of autosar embedded applications. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, pages 384–390, 2014.
25. Brian Dougherty, Jules White, Jaiganesh Balasubramanian, Chris Thompson, and Douglas C Schmidt. Deployment automation with blitz. In *2009 31st International Conference on Software Engineering-Companion Volume*, pages 271–274. IEEE, 2009.
26. Salah Eddine Saidi, Sylvain Cotard, Khaled Chaaban, and Kevin Marteil. An ilp approach for mapping autosar runnables on multi-core architectures. In *Proceedings of the 2015 Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools*, pages 1–8, 2015.
27. Kabsu Han, Daejin Park, and Jeonghun Cho. An fds algorithm for synthesis of autosar architecture. *Advanced ence Letters*, 23(3):1608–1612, 2017.
28. Jaeho Park and Byoung Wook Choi. Design and implementation procedure for an advanced driver assistance system based on an open source autosar. *Electronics*, 8(9):1025, 2019.
29. Cristian Spirleanu and Eugen Diaconescu. Application model in autosar software development for control systems design through fuzzy methods. In *SIAR International Congress of Automotive and Transport Engineering: Science and Management of Automotive and Transportation Engineering*, pages 508–517. Springer, 2019.
30. K Senthilkumar and Ramesh Ramadoss. Optimized scheduling of multicore ecu architecture with bio-security can network using autosar. *Future Generation Computer Systems*, 98:1–11, 2019.
31. Masoud Rabbani, Alireza Nikoubin, and Hamed Farrokhi-Asl. Using modified metaheuristic algorithms to solve a hazardous waste collection problem considering workload balancing and service time windows. *Soft Computing*, pages 1–28, 2020.
32. Chung Laung Liu and James W Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, 20(1):46–61, 1973.
33. Anne Condon and Richard M Karp. Algorithms for graph partitioning on the planted partition model. *Random Structures & Algorithms*, 18(2):116–140, 2001.