IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 9, 2020, accepted June 7, 2020, date of publication June 15, 2020, date of current version June 25, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3002335

Guarantee the Quality-of-Service of Control Transactions in Real-Time Database Systems

CHENGGANG DENG[1], GUOHUI LI[2], QUAN ZHOU[1], AND JIANJUN LI[1]
[1]Department of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
[2]Department of Software Engineering, Huazhong University of Science and Technology, Wuhan 430074, China

Corresponding author: Quan Zhou (quanzhou@hust.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61672252 and Grant 61802137, in part by the Fundamental Research Funds for the Central Universities under Grant 2019kfyXKJC021, and in part by the Hubei Provincial Natural Science Foundation of China under Grant 2018CFB204.

ABSTRACT Temporal consistency analysis is an important problem in Real-Time Database Systems (RTDBSs). Most of existing works on hybrid transaction co-scheduling focus on exploring effective deadline and period assignment methods which can guarantee the temporal consistency of data objects and the schedulability of transactions. To the best of our knowledge, all existing researches have not yet invested sufficient efforts on guaranteeing the Quality of Service (QoS) of control transactions and allow control transactions finish their execution after the data obtained by these control transactions expire. In this paper, we define a new problem of how to schedule a hybrid transaction set under the guarantee on QoS of control transactions. Two deadline and period assignment methods for addressing the defined problem are proposed in this work. Both the proposed methods support the Earliest Deadline First (EDF) scheduling. Extensive experiments are conducted to evaluate the performances of the proposed methods in terms of acceptance ratio, processor workload and efficiency.

INDEX TERMS Real-time database systems, QoS of control transactions, hybrid transaction set, the EDF scheduling, deadline and period assignment.

I. INTRODUCTION

Real-time database systems (RTDBSs) have been widely applied in many areas. Examples of such applications include industrial control [1], vehicular control [2], aerospace control [3], health monitoring [4] and robot control [5]. There are many transactions in a RTDBS. The RTDBS schedules and executes these transactions for guaranteeing that all transactions can finish their execution before their absolute deadlines. The Earliest Deadline First (EDF) scheduling [6] and the Fixed Priority (FP) scheduling [7] are two kinds of representative scheduling methods. Each transaction has a fixed priority under FP. But, under EDF, transactions with earlier relative deadlines have higher priorities. It has been proved that EDF dominates FP in uni-processor systems [8] and EDF outperforms DM for the temporal consistency maintenance problem from the perspective of success ratio and processor workload [9]. Therefore, we consider a uni-processor RTDBS with EDF in this work.

The associate editor coordinating the review of this manuscript and approving it for publication was Zhiwu Li.

Transactions in RTDBSs can be classified into two kinds: control transactions and update transactions. Control transactions access data objects (in RTDBSs) and perform corresponding operations according to the values they get. Update transactions are responsible for monitoring external objects and updating data objects (in RTDBSs) according to the states of the objects they monitor. The deadlines and the periods of control transactions are usually determined but those of update transactions are usually undetermined. Most of existing works for scheduling transactions in RTDBSs assume that each value of each data object has a valid time interval in which this value is valid, and RTDBSs need to assign deadlines and periods for control transactions and schedule transactions under the guarantee that the values of all data objects in RTDBSs are valid at arbitrary time instant [10]–[15]. However, only guaranteeing the freshness of data objects (in RTDBSs) is not enough in some special applications and RTDBSs need to guarantee the cleanliness of the execution results of control transactions. This means each value of each data object may have a longest service life and all control transactions need to finish their

execution before the values they obtain exceed the corresponding longest service lives. Consider the Light Detection And Ranging (LiDAR) module in the unmanned vehicle system. Since the dynamic detection time for each frame of data is about 30 milliseconds [16] and it needs at least two to three frames from detection to stable target tracking, the system needs to assign some reasonable deadlines and Longest Service Lives (LSLs) for control transactions to guarantee that each decision can be made within 100 milliseconds. Clearly, although the existing methods can guarantee the freshness of data objects, they cannot be used in these applications.

In this work, we study the co-scheduling problem in the RTDBSs with strict requirements on the service lives of the values of data objects. The main contributions are summarized as bellows.

1) A new problem is defined in this paper, in which the service lives of the values of data objects are considered and control transactions are required to finish their execution before the values they obtain expire.
2) Two effective co-scheduling methods (named **minD*-SLG** and **DPR-SLG**, respectively) for the defined problem are proposed. **minD*-SLG** develops the **minD*** method to resolve new problem. **DPR-SLG** improves **minD*-SLG** on the acceptance ratio by readjusting the deadlines and the periods of update transactions if transactions are unschedulable under **minD*-SLG**. The improvement comes at the price of efficiency.
3) A set of experiments is conducted to evaluate the performances of the proposed methods. The result shows that both algorithms can reduce processor load by about 25%, **DPR-SLG** can enhance the acceptance ratio of **minD*-SLG** for about 700% and the average execution time of **DPR-SLG** is about 1.5 times of that of **minD*-SLG**.

Organization: The rest of this paper is organized as follows: Section II discusses the related works. Section III reviews the definition of temporal validity of data objects and describes the notations used in this paper, as well as gives the definition of the problem studied in this work. Two effective co-scheduling methods for the defined problem are proposed in Section IV. Section V presents the performances of the proposed methods in experiments, followed by the conclusion in Section VI.

II. RELATED WORK

There has been a lot of work about guaranteeing the QoS of task scheduling [17]–[21]. Bi *et al.* [17] proposed a new approach to optimize the profit of virtualized cloud data centers between service providers and customers. Yuan *et al.* [18] considered the cost minimization of private cloud data centers in hybrid clouds and proposed the Temporal Task Scheduling Algorithm (*TTSA*) to effectively allocate all arriving tasks to private cloud data centers and public clouds. Yuan *et al.* [19] studied the problem of how to maximize the profit of a private cloud in hybrid clouds while guaranteeing the service delay bound of delay-tolerant tasks. Yuan *et al.* [20] first proposed a revenue-based workload admission control method for selectively accepting requests and proposed a cost-aware workload scheduling method to distribute requests among multiple available internet service providers connected to distributed cloud data centers. Yuan *et al.* [21] proposed a Time-Aware Task Scheduling (TATS) algorithm to investigate the temporal variation and schedule all admitted tasks to execute in green data centers meeting their delay bounds.

For RTDBSs, maintaining the freshness of data is an important aspect of guaranteeing the QoS of RTDBSs. There has been a lot of work for maintaining the freshness of real-time data objects [2], [15], [22]–[26]. Song and Liu [27] studied the performances of the two-phase locking and the optimistic algorithm in maintaining temporal consistency of shared data in hard real-time systems with periodic tasks. Kuo and Mok [28] investigated real-time data-semantics and proposed a class of real-time access protocal named Similarity Stack Protocal (*SSP*). Ho *et al.* [13] proposed a semantics-based reconfiguration method by combining the Half-Half (*HH*) scheme with similarity-based principles to reduce workload by skipping the execution of transaction instances. The proposed method realizes the balance between data consistency and processor workload. Xiong *et al.* [29] introduced the concept of data-deadline and proposed a data-deadline based scheduling algorithm, as well as forced-wait and similarity based scheduling techniques, to maintain the temporal consistency of real-time data. Gustafsson and Hansson [2] proposed the On-Demand Top-Bottom traversal with relevance check (*ODTB*) method for updating data items that can skip unnecessary updates allowing for better CPU utilization. The deadlines and periods of update transactions are assumed to have been assigned in the above works.

Various deadline and period deriving methods for maintaining the temporal consistency are proposed in recent two decades. To address the temporal consistency maintenance problem, Xiong and Ramamritham [15] proposed the More-Less (*ML*) scheme in which all transactions are released periodically under DM scheduling. Xiong *et al.* [30] proposed the Deferrable Scheduling algorithm for Fixed Priority transactions (*DS-FP*) which supports the sporadic transaction model and reduces processor workload by judiciously deferring the sampling times of update transaction jobs. Han *et al.* [31] proposed the *DS-EDF* method which extends *DS-FP* to EDF scheduling environments. The schedulability of *DS-EDF* is much worse than *DS-FP* in some cases, though the total update workload from *DS-EDF* has been shown to be slightly lower than *DS-FP*. Jha *et al.* [23] investigated formulas that give the maximal value of mutual gaps among a set of data reads and proposed the first deadline and period deriving method which can guarantee the mutual temporal consistency of real-time data objects. Han *et al.* [32] proposed two online scheduling switch schemes, Search-Based Switch (*SBS*) and Adjustment-Based Switch (*ABS*), to search for the proper switch point to maintain the temporal validity during the mode changes. Zhu *et al.* [33] proposed a linear

programming based method to address the deadline calculation problem when there are only update transactions scheduled under EDF scheduling. All above methods can be used in the environments with only update transactions or control transactions.

Recently, the co-scheduling of hybrid transactions becomes a hot research topic in RTDBSs. Han *et al.* [34] proposed a real-time co-scheduling algorithm, called Adaptive Earliest Deadline First Co-Scheduling (*AEDF-Co*) method, with the objective of determining a schedule for a given hybrid transaction set so that the deadline constraints of all the control transactions are satisfied while the quality of data (*QoD*) of the real-time data objects is maximized. Wang *et al.* [35] proposed the Periodic Co-Scheduling (*PCS*) method that adopts a Fix Priority (FP) assignment scheme to maintain the temporal validity of real-time data objects. Han *et al.* [12] proposed a co-scheduling algorithm called deferrable Co-scheduling with Least Actual Laxity First (*Co-LALF*), in which the release times of update jobs are deferred for reducing the process workload. All the above three methods aim to maximize the *QoD* under the premise of the schedulability of control transactions, thus update transactions may miss their deadlines. Li *et al.* [36] considered the problem of how to derive deadline and period for update transactions to maintain the temporal consistency of real-time data objects, while guaranteeing the hybrid transaction set to be EDF-schedulable. However, this work does not guarantee the cleanliness of the execution results of control transactions.

Different from the above, we focus on guaranteeing the QoS of control transactions and allow control transactions finish their execution after the data obtained by these control transactions expire. So, our work can be considered to be complementary to theirs.

III. NOTATIONS AND PROBLEM DEFINITION

We first review the definition of the temporal validity of data objects and give some useful notations. Then, we define the problem to be addressed in this work.

A. TEMPORAL VALIDITY OF DATA OBJECTS

Data objects in RTDBSs are used to record the current states of external objects. Since the states of external objects change over time, it is necessary to update these real-time data objects timely for guaranteeing their temporal validity.

Definition 1: [37] At a time instant t, a real-time data object is temporally valid if the sum of its latest sampling time and its validity interval length is no smaller than t.

B. NOTATIONS

A uni-processor RTDBS with n data objects and $2n$ hybrid transactions is considered in this work. We assume that transactions are scheduled by EDF, and each data object is updated by only one update transaction and accessed by only one control transaction. So, the transaction set consists of n update transactions and n control transactions.

$\mathcal{O} = \{o_1, o_2, \ldots, o_n\}$ is used to denote the data object set in the RTDBS and o_i $(1 \leq i \leq n)$ is the i-th data object in $\mathcal{O}$. Each data object o_i has a validity interval length $\mathcal{V}_i$. Use $\mathcal{V}_{max}$ and $\mathcal{V}_{min}$ to denote the maximum and minimun validity interval length in $\mathcal{O}$. Use $S_{i,l}$ to denote the l-th sampled result of o_i and use $t_{i,l}$ to denote the sampled time of $S_{i,l}$. $S_{i,l}$ is fresh only in $[t_{i,l}, t_{i,l} + \mathcal{V}_i)$. Each sampled result $S_{i,l}$ has a LSL, θ_i. The control transaction getting $S_{i,l}$ needs to finish its execution before $t_{i,l} + \theta_i$.

$\mathcal{T}^u = \{\tau_1^u, \tau_2^u, \ldots, \tau_n^u\}$ is used to denote the update transaction set. The i-th transaction in $\mathcal{T}^u$, τ_i^u, is the update transaction that is responsible for updating o_i. Each update transaction can be characterized by a 3-tuple: $< C_i^u, D_i^u, T_i^u >$, where C_i^u is the worst case execution time, D_i^u is the relative deadline and T_i^u is the period. Since deadline-constrained transactions are considered in this work, we require that $D_i^u \leq T_i^u$ for each τ_i^u. Using $U_i^u = C_i^u/T_i^u$ to denote the utilization of τ_i^u, the total utilization of $\mathcal{T}^u$, U_{total}^u, can be obtained by $U_{total}^u = \sum_{i=1}^{n} U_i^u$. The values of D_i^u and T_i^u is undetermined before the deadline and period assignment for τ_i^u. But, it is required that $D_i^u + T_i^u \leq \mathcal{V}_i$. Each update transaction can generate an infinite stream of jobs. Use $J_{i,j}^u$ to denote the j-th job of τ_i^u. $r_{i,j}^u$ and $d_{i,j}^u$ are the release time and the absolute deadline of $J_{i,j}^u$, respectively, where $d_{i,j}^u = r_{i,j}^u + D_i^u$. Each update job $J_{i,j}^u$ takes sample of o_i at its release time $r_{i,j}^u$, which means $r_{i,j}^u$ is the j-th sampled time of o_i.

$\mathcal{T}^c = \{\tau_1^c, \tau_2^c, \ldots, \tau_n^c\}$ is used to denote the control transaction set in the RTDBS and τ_i^c is the i-th transaction in $\mathcal{T}^c$. Each control transaction τ_i^c can be characterized by a 3-tuple: $< C_i^c, D_i^c, T_i^c >$, where C_i^c is the worst case execution time, D_i^c is the relative deadline and T_i^c is the period. Similar to update transactions, we require that $D_i^c \leq T_i^c$. We assume that τ_i^c accesses o_i and performs corresponding operations according to the value it gets. $S_{i,k}^c$ is used to denote the value (of o_i) obtained by a control job $J_{i,k}^c$ and $t_{i,k}^c$ is used to denote the sample time of $S_{i,k}^c$. Moreover, using U_i^c and U_{total}^c to denote the utilization of τ_i^c and the total utilization of $\mathcal{T}^c$, respectively, we have $U_i^c = C_i^c/T_i^c$ and $U_{total}^c = \sum_{i=1}^{n} U_i^c$. U_{total} is denoted as the total utilization of $\mathcal{T}$, where $U_{total} = U_{total}^c + U_{total}^u$. Each control transaction τ_i^c can generate an infinite stream of jobs and $J_{i,k}^c$ is used to denote the k-th job generated by τ_i^c. Using $r_{i,k}^c$, $d_{i,k}^c$ and $f_{i,k}^c$ to denote the release time, the absolute deadline and the finish time of $J_{i,k}^c$, respectively, the response time of $J_{i,k}^c$ (denoted by $R_{i,k}^c$) can be obtained by $R_{i,k}^c = f_{i,k}^c - r_{i,k}^c$ and the Worst-Case Response Time (WCRT) of τ_i (denoted by R_i^c) is the maximum $R_{i,k}^c$, $k \geq 1$. Different from update transactions, the deadlines and the periods of control transactions are determined in this work.

All useful notations are shown in Table 1.

C. PROBLEM TO BE ADDRESSED

A new problem is defined in this section. Before defining the new problem, we review an existing problem, the DPC problem [36], defined as follows.

***Deadline and Period Calculation (DPC) problem**: [36] Given a hybrid transaction set $\mathcal{T}$ which consists of update*

TABLE 1. Notations in this paper.

$\mathcal{O}$	external object set	o_i	the i-th external object
$\mathcal{V}_i$	valid interval length of o_i	$S_{i,l}$	the l-th sample of o_i
$t_{i,l}$	sampling time of o_i	θ_i	LSL of each sample of o_i
$\mathcal{T}^u$	update transaction set	τ_i^u	the i-th update transaction
C_i^u	WCET of τ_i^u	D_i^u	relative deadline of τ_i^u
T_i^u	period of τ_i^u	U_i^u	utilization of τ_i^u
U_{total}^u	total utilization of $\mathcal{T}^u$	$J_{i,j}^u$	the j-th job of τ_i^u
$r_{i,j}^u$	release time of $J_{i,j}^u$	$d_{i,j}^u$	absolute deadline of $J_{i,j}^u$
$\mathcal{T}^c$	control transaction set	τ_i^c	the i-th control transaction
C_i^c	WCET of τ_i^c	D_i^c	relative deadline of τ_i^c
T_i^c	period of τ_i^c	U_i^c	utilization of τ_i^c
U_{total}^c	total utilization of $\mathcal{T}^c$	$J_{i,j}^c$	the j-th job of τ_i^c
$r_{i,j}^c$	release time of $J_{i,j}^c$	$d_{i,j}^c$	absolute deadline of $J_{i,j}^c$
$f_{i,j}^c$	finish time of $J_{i,j}^c$	U_{total}	total utilization of $\mathcal{T}$
$S_{i,j}^c$	data $J_{i,j}^c$ gets	$t_{i,j}^c$	sampling time of $S_{i,j}^c$
$R_{i,j}^c$	response time of $J_{i,j}^c$	R_i^c	WCRT of τ_i^c

transaction set $\mathcal{T}^u$ and control transaction set $\mathcal{T}^c$, derive deadlines and periods for the update transactions in $\mathcal{T}^u$, such that,

1) *$\mathcal{T}$ is schedulable under EDF.*
2) *The total workload of $\mathcal{T}$ is minimized.*

Different from the DPC problem, each data object o_i has a LSL, θ_i, in the new problem. Each control job needs to finish its execution before its obtained data object exceeds the service life. Specifically, the new problem is defined as follows.

***Co-Scheduling with Service Life Guarantee of data objects (CS-SLG)**: Given a external object set $\mathcal{O}$ and a hybrid transaction set $\mathcal{T}$ consisting of a control transaction set $\mathcal{T}^c$ and an update transaction set $\mathcal{T}^u$, assign deadlines and periods for update transactions (in $\mathcal{T}^u$) and schedule transactions (in $\mathcal{T}$), such that:*

1) *$\mathcal{T}$ is schedulable under EDF.*
2) *The total utilization of $\mathcal{T}$ is minimized.*
3) *$D_i^u + T_i^u = \mathcal{V}_i$ for each update transaction τ_i^u.*
4) *$f_{i,k}^c - t_{i,k}^c \le \theta_i$ for each control job $J_{i,k}^c$.*

IV. TWO EFFECTIVE METHODS FOR THE CS-SLG PROBLEM

In this section, we propose two effective methods for the CS-SLG problem. Since our proposed methods are effective under EDF scheduling, both the job preemption and the job priorities follow the corresponding rules in EDF. In addition, our proposed methods are executed before the system executes transactions and would not be executed in parallel with the transactions in the transaction set. Thus, the execution of the algorithm will not be affected by the preemption between jobs in the transaction set.

A. THE MIND*-SLG Method

minD* [36] is an existing deadline and period assignment method for the DPC problem. In this section, we propose an effective co-scheduling method, minD*-SLG, which extends minD* to resolve the CS-SLG problem.

Before showing our minD*-SLG method, we give a definition as follows.

Definition 2: The QoS of $J_{i,k}^c$ can be guaranteed if $J_{i,k}^c$ finishes its execution before $t_{i,k}^c + \theta_i$ (i.e., $f_{i,k}^c \le t_{i,k} + \theta_i$). The QoS of a control transaction τ_i^c can be guaranteed if the QoS of all jobs generated by τ_i^c can be guaranteed.

Next, we give a useful lemma as follows.

Lemma 1: Given an EDF-schedulable hybrid transaction set $\mathcal{T}$ and four jobs $J_{i,k}^c$, $J_{i,j}^u$, $J_{i,j+1}^u$ and $J_{i,j+2}^u$ (generated by transactions in $\mathcal{T}$), the sample time of $S_{i,k}^c$, $t_{i,k}^c$, satisfies $t_{i,k}^c \ge r_{i,j}^u$ if $r_{i,j+1}^u \le r_{i,k}^c < r_{i,j+2}^u$.

Proof: Since $D_i^u \le T_i^u$ and $r_{i,j+1}^u \le r_{i,k}^c < r_{i,j+2}^u$, we have $d_{i,j}^u = r_{i,j}^u + D_i^u \le r_{i,j} + T_i^u \le r_{i,j+1}^u \le r_{i,k}^c$, which means $J_{i,j}^u$ has completed the update of o_i before $J_{i,k}^c$ accesses o_i. Therefore, $J_{i,k}^c$ must get a data updated by $J_{i,j}^u$ or $J_{i,j+1}^u$. Since $r_{i,j}^u$ is the sample time of o_i by $J_{i,j}^u$, we have $t_{i,k}^u \ge r_{i,j}^u$. □

Since the QoS of $J_{i,k}^c$ can be guaranteed if $f_{i,k}^c \le t_{i,k}^c + \theta_i$, $f_{i,k}^c \le r_{i,j}^u + \theta_i$ is a sufficient condition for guaranteeing the QoS of $J_{i,k}^c$ based on Lemma 1. Note that all transactions are released periodically and 0 is the time instant at which all transactions are released for the first time. Since $r_{i,j+1}^u \le r_{i,k}^c < r_{i,j+2}^u$, we can obtain that

$$\begin{aligned} r_{i,j}^u &= r_{i,j+1}^u - T_i^u \\ &= \lfloor r_{i,k}^c / T_i^u \rfloor T_i^u - T_i^u \\ &= \lfloor (k-1)T_i^c / T_i^u \rfloor T_i^u - T_i^u \\ &= (\lfloor (k-1)T_i^c / T_i^u \rfloor - 1)T_i^u, \end{aligned} \tag{1}$$

which means $f_{i,k}^c \le (\lfloor (k-1)T_i^c/T_i^u \rfloor - 1)T_i^u + \theta_i$ (i.e., Equation (2)) is a sufficient condition for guaranteeing the QoS of $J_{i,k}^c$.

$$R_{i,k}^c \le (\lfloor (k-1)T_i^c/T_i^u \rfloor - 1)T_i^u + \theta_i - (k-1)T_i^c \tag{2}$$

Based on the above analysis, we can get an important theorem as follows.

Theorem 1: Given an EDF-schedulable transaction set $\mathcal{T} = \{\tau_1^c, \ldots, \tau_n^c, \tau_1^u, \ldots, \tau_n^u\}$, the QoS of a control transaction τ_i^c can be guaranteed if the WCRT of this transaction, R_i^c, satisfies Equation (3).

$$R_i^c \le \theta_i - 2T_i^u \tag{3}$$

Proof: Since R_i^c is the maximum $R_{i,k}^c$ ($k \ge 1$), we can get that $R_{i,k}^c \le R_i^c$ for each $k \ge 1$. Since R_i^c satisfies Equation (3), $R_{i,k}^c \le \theta_i - 2T_i^u$ for each $k \ge 1$. Note that,

$$\begin{aligned} &(\lfloor (k-1)T_i^c/T_i^u \rfloor - 1)T_i^u + \theta_i - (k-1)T_i^c \\ &= \theta_i - T_i^u - (k-1)T_i^c + \lfloor (k-1)T_i^c/T_i^u \rfloor T_i^u \\ &\ge \theta_i - T_i^u - (k-1)T_i^c + ((k-1)T_i^c/T_i^u - 1)T_i^u \\ &\ge \theta_i - T_i^u - (k-1)T_i^c + (k-1)T_i^c - T_i^u \\ &\ge \theta_i - 2T_i^u \end{aligned} \tag{4}$$

So, $R_{i,k}^c \le (\lfloor (k-1)T_i^c/T_i^u \rfloor - 1)T_i^u + \theta_i - (k-1)T_i^c$ for each $k > 1$, which means all $R_{i,k}^c$s satisfy Equation (2) and

Algorithm 1 The minD*-SLG method

input : A hybrid transaction set $\mathcal{T} = \mathcal{T}^u \bigcup \mathcal{T}^c$.
output: The hybrid transaction set after the deadline and period assignment for update transactions.

1 **begin**
2 Let $T_i^u = D_i^u = \mathcal{V}_i/2$ for each update transaction τ_i^u;
3 Use **QPA** to test the EDF-schedulability of $\mathcal{T}$;
4 **if** $\mathcal{T}$ *is unschedulable under EDF* **then**
5 Return "Fail";
6 **else**
7 Sort update transactions in the non-decreasing order of their deadlines;
8 Use the **minD*** method to reassign the deadline and the period of each update transaction;
9 $flag = 1$;
10 **for** $i = 1; i \leq n; i++$ **do**
11 Calculate the WCRT of τ_i^c, R_i^c;
12 **if** R_i^c *does not satisfy Equation (3)* **then**
13 $flag = 0$; break;
14 **if** $flag == 1$ **then**
15 Return $\mathcal{T}$;
16 **else**
17 Return "Fail";

the QoS of all jobs generated by τ_i^c can be guaranteed. Based on Definition 2, we can derive that the QoS of τ_i^c can be guaranteed. The proof thus finishes. □

Theorem 1 gives a sufficient condition for guaranteeing the QoS of a control transaction τ_i^c. Since **minD*** can realize the deadline and period assignment for update transactions (in an EDF-schedulable hybrid transaction set), we can derive an effective methods (named **minD*-SLG**) for the **CS-SLG** problem by combining the theory of Theorem 1 with the **minD*** method.

Algorithm 1 shows the pseudo code of our **minD*-SLG** method. As shown in Algorithm 1, the **minD*-SLG** method first sets $D_i^u = T_i^u = \mathcal{V}_i/2$ for each update transaction τ_i^u (at Line 2). Then, **QPA** [38] is used to test the schedulability of $\mathcal{T}$ (at Line 3). If the result is "unschedulable", we return a "Fail" result (at Line 5). Otherwise, all update transactions are sorted in the non-decreasing order of their deadlines (at Line 7) and the **minD*** method is used to reassign the deadlines and periods of update transactions (at Line 8). Next, *flag* is initialized to 1 (at Line 9) and the WCRT of each control transaction is calculated (at Line 11). It should be pointed out that the existing RTA methods can be used for calculating the WCRT of τ_i^c in **minD*-SLG**. Once there appears an R_i^c unsatisfying Equation (3), we let $flag = 0$ and finish the calculation of R_i^cs (at Line 13). Finally, the schedule $\mathcal{T}$ is returned (at Line 15) if all R_i^cs satisfy Equation (3) (i.e., $flag == 1$). Otherwise, **minD*-SLG** returns a "Fail" result (at Line 17). Since executing **QPA** once takes at most $O(n \times L)$ time, where L is the maximal busy period size. Besides, the overall complexity of **minD*** to a periodic task set is $O(kn^2)$, where $k = \max\{\frac{\mathcal{V}_i}{2} - \max(C_i^u, D_{i-1}^u)\}_{i=1}^n$. Thus, the overall complexity of Algorithm 1 is $O(n \times (kn + L + TL^2))$.

Note that, although **minD*-SLG** is an effective co-scheduling method for the **CS-SLG** problem, its acceptance ratio is unsatisfactory since the deadlines and the periods of update transactions are determined before the QoS test on control transactions. Therefore, the acceptance ratio of **minD*-SLG** can be improved by readjusting the deadlines and the periods of update transactions after the QoS test on control transactions.

B. THE DPR-SLG METHOD

As described above, **minD*-SLG** has serious drawback in terms of acceptance ratio. In this section, we proposed a new algorithm, named Deadline and Period Reassignment for Service Life Guarantee (**DPR-SLG**), which has a higher acceptance ratio than **minD*-SLG**.

First, we give an important theorem as follows.

*Theorem 2: If $R_i^c + 2T_i^u > \theta_i$ after the deadline and period assignment by **minD***, $\overline{R_i^c} + 2\overline{T_i^u} > \theta_i$ after extending D_i^u to $\overline{D_i^u} = D_i^u + m$, where $\overline{T_i^u} = \mathcal{V}_i - \overline{D_i^u}$, $\overline{R_i^c}$ is the new WCRT of τ_i^c, m, $\mathcal{P}$ and $\mathcal{Q}$ satisfy Equations (5), (6), (7) and (8), respectively.*

$$m = \max\{0, \left\lceil T_i^u - \mathcal{P} - \sqrt{\mathcal{P}^2 - \frac{\mathcal{Q} \cdot C_i^u}{2}} \right\rceil - 1\} \tag{5}$$

$$\sqrt{\mathcal{P}^2 - \frac{\mathcal{Q} \cdot C_i^u}{2}} \geq 0 \tag{6}$$

$$\mathcal{P} = \frac{1}{4}[(1 + \frac{R_i^c}{T_i^u})C_i^u + \theta_i - R_i^c] \tag{7}$$

$$\mathcal{Q} = R_i^c - T_i^u + 1 \tag{8}$$

Proof: Considering an arbitrary control job $J_{i,k}^c$ and using $J_{i,j+1}^u$ to denote the update job satisfying $r_{i,j+1}^u \leq r_{i,k}^c < r_{i,j+2}^u$, we first prove that $\overline{R_i^c} + 2\overline{T_i^u} > \theta_i$, where $\overline{T_i^u} = \mathcal{V}_i - \overline{D_i^u}$.

Use $\overline{R_i^c}$ to denote the new WCRT of τ_i^c (after the extension of D_i^u) and let $\Delta_i = R_i^c + 2T_i^u - \theta_i$ and $\overline{\Delta_i} = \overline{R_i^c} + 2\overline{T_i^u} - \theta_i$. Since all existing Response Time Analysis (RTA) methods assume that higher priority jobs are released periodically, the release times of τ_i^u in every time interval with length R_i^c is equal to $\left\lceil \frac{\mathcal{Q}}{T_i^u} \right\rceil$ or $\left\lceil \frac{R_i^c}{T_i^u} \right\rceil$ under D_i^u, and that under $\overline{D_i^c} = D_i^u + m$ is equal to $\left\lceil \frac{\mathcal{Q}}{T_i^u} \right\rceil$ or $\left\lceil \frac{R_i^c}{T_i^u} \right\rceil$. Figures 1 and 2 show the scenarios in which these release times can be obtained.

Based on the above analysis, we can derive that the number of the jobs (of τ_i^u) with both release times and end times in $[s_0, s_1]$ will be added for at least $\left\lceil \frac{\mathcal{Q}}{T_i^u} \right\rceil - \left\lceil \frac{R_i^c}{T_i^u} \right\rceil$, which means the interference on a job $J_{i,k}^c$ in $[r_{i,k}^c, r_{i,k}^c + R_i^c)$ will increase for at least $(\left\lceil \frac{\mathcal{Q}}{T_i^u} \right\rceil - \left\lceil \frac{R_i^c}{T_i^u} \right\rceil)C_i^u$. Since we consider a uni-processor

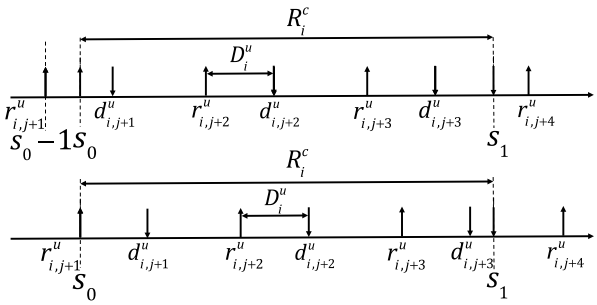


FIGURE 1. Two cases of τ_i^u in $[s_0, s_1)$ under D_i^u.

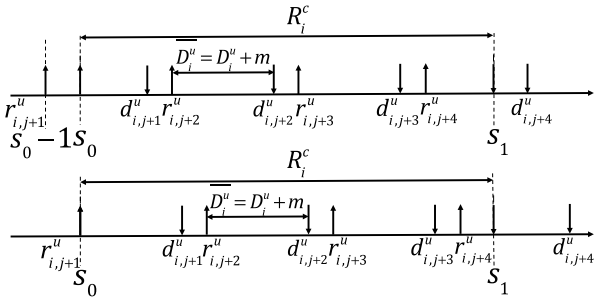


FIGURE 2. Two cases of τ_i^u in $[s_0, s_1)$ under $\overline{D_i^u} = D_i^u + m$.

system in this work, it can be derived that $\overline{R_i^c} - R_i^c \geq (\left\lceil \frac{\mathcal{Q}}{\overline{T_i^u}} \right\rceil - \left\lceil \frac{R_i^c}{T_i^u} \right\rceil) C_i^u$. By $\Delta_i = R_i^c + 2T_i^u - \theta_i$ and $\overline{\Delta_i} = \overline{R_i^c} + 2\overline{T_i^u} - \theta_i$, we have

$$\begin{aligned}
&\overline{\Delta_i} - \Delta_i \\
&= \overline{R_i^c} - R_i^c + 2(\overline{T_i^u} - T_i^u) \\
&= \overline{R_i^c} - R_i^c - 2m \\
&\geq (\left\lceil \frac{\mathcal{Q}}{\overline{T_i^u}} \right\rceil - \left\lceil \frac{R_i^c}{T_i^u} \right\rceil) C_i^u - 2m \\
&\geq (\frac{\mathcal{Q}}{\overline{T_i^u}} - (\frac{R_i^c}{T_i^u} + 1)) C_i^u - 2m \\
&\geq (\frac{\mathcal{Q}}{\overline{T_i^u}} - \frac{R_i^c}{T_i^u} - 1) C_i^u - 2m
\end{aligned} \tag{9}$$

So, $\overline{\Delta_i} > 0$ if $\Delta_i + (\frac{\mathcal{Q} - \overline{T_i^u}}{\overline{T_i^u}} - \frac{R_i^c}{T_i^u}) C_i^u - 2m > 0$. Since $\overline{T_i^u} > 0$, we can get that $\overline{\Delta_i} > 0$ if $f(\overline{T_i^u}) = \Delta_i \overline{T_i^u} + (\mathcal{Q} - \overline{T_i^u} - \frac{R_i^c}{T_i^u} \overline{T_i^u}) C_i^u - 2(T_i^u - \overline{T_i^u}) \overline{T_i^u} > 0$.

Note that, $f(\overline{T_i^u})$ satisfies

$$\begin{aligned}
&f(\overline{T_i^u}) \\
&= \Delta_i \overline{T_i^u} + (\mathcal{Q} - \overline{T_i^u} - \frac{R_i^c}{T_i^u} \overline{T_i^u}) C_i^u - 2(T_i^u - \overline{T_i^u}) \overline{T_i^u} \\
&= 2\overline{T_i^u}^2 + (\Delta_i - C_i^u - \frac{R_i^c}{T_i^u} C_i^u - 2T_i^u) \overline{T_i^u} + \mathcal{Q} \cdot C_i^u \\
&= 2\overline{T_i^u}^2 + (R_i^c - \theta_i - (1 + \frac{R_i^c}{T_i^u}) C_i^u) \overline{T_i^u} + \mathcal{Q} \cdot C_i^u \\
&= 2\overline{T_i^u}^2 - 4\mathcal{P}\overline{T_i^u} + \mathcal{Q} \cdot C_i^u \\
&= 2(\overline{T_i^u} - \mathcal{P})^2 - (2\mathcal{P}^2 - \mathcal{Q} \cdot C_i^u)
\end{aligned} \tag{10}$$

Algorithm 2 The DPR-SLG method

input : A hybrid transaction set $\mathcal{T} = \mathcal{T}^u \bigcup \mathcal{T}^c$.
output: The hybrid transaction set after the deadline and period assignment for update transactions.

1 **begin**
2 Let $T_i^u = D_i^u = \mathcal{V}_i/2$ for each update transaction τ_i^u;
3 Utilize **QPA** to judge the EDF-schedulability of $\mathcal{T}$;
4 **if** $\mathcal{T}$ *is unschedulable under EDF* **then**
5 Return "Fail";
6 **else**
7 Sort update transactions in the non-decreasing order of their deadlines;
8 Use the **minD*** method to reassign the deadline and the period of each update transaction;
9 **for** $i = 1; i \leq n; i++$ **do**
10 Calculate the WCRT of τ_i^c, R_i^c;
11 **while** R_i^c *does not satisfy Equation (3) and* $D_i^u \leq \mathcal{V}_i/2$ **do**
12 Let m satisfy Equation (5);
13 $D_i^u = D_i^u + m + 1$;
14 $T_i^u = \mathcal{V}_i - D_i^u$;
15 Calculate the WCRT of τ_i^c, R_i^c;
16 **if** $D_i^u > \mathcal{V}_i/2$ **then**
17 Return "Fail";
18 Return $\mathcal{T}$;

Therefore, $f(\overline{T_i^u}) > 0$ if $\overline{T_i^u} > \mathcal{P} + \sqrt{\mathcal{P}^2 - \frac{\mathcal{Q} \cdot C_i^u}{2}}$ or $\overline{T_i^u} < \mathcal{P} - \sqrt{\mathcal{P}^2 - \frac{\mathcal{Q} \cdot C_i^u}{2}}$. Since $\overline{T_i^u} = T_i^u - m, m = T_i^u - \overline{T_i^u} < T_i^u - \mathcal{P} - \sqrt{\mathcal{P}^2 - \frac{\mathcal{Q} \cdot C_i^u}{2}}$ is a sufficient condition for guaranteeing $f(\overline{T_i^u}) > 0$ (also $\overline{\Delta_i} > 0$). Since m is a nonnegative integer, we can derive that m satisfies Equation (5) is a sufficient condition for guaranteeing $\overline{\Delta_i} > 0$ (i.e., $\overline{R_i^c} + 2\overline{T_i^u} > \theta_i$). The proof thus finishes. □

Note that, $R_i^c + 2T_i^u \leq \theta_i$ is a sufficient condition for guaranteeing the QoS of τ_i^c (based on Theorem 1). So, $\overline{D_i^u} = D_i^u + m + 1$ is a lower bound of the deadline of τ_i for guaranteeing the QoS of τ_i^c if $R_i^c + 2T_i^u > \theta_i$ (based on Theorem 2). Thus, we can improve **minD*-SLG** by extending the deadline of τ_i^c, D_i^c, to $D_i^c + m + 1$ if $R_i^c + 2T_i^u > \theta_i$ after the deadline assignment for τ_i^c. Based on the above idea, we propose a new co-scheduling method (**DPR-SLG**). The pseudo-code is shown in Algorithm 2.

Different from **minD*-SLG**, **DPR-SLG** reassigns the deadline of τ_i^u to $D_i^u + m + 1$ and recalculates the WCRT of τ_i^c if Equation (3) cannot be met by τ_i^c under D_i^u (at Lines 12 to 14). The above process is repeated until τ_i^c satisfies Equation (3) or $D_i^u > \mathcal{V}_i/2$. If $D_i^u > \mathcal{V}_i/2$, a "Fail" result is returned (at Line 17). Otherwise, **DPR-SLG** continues to assign the deadline for the next update transaction and returns the final schedule $\mathcal{T}$ (at Line 18) if all update transactions

TABLE 2. Transactions in Example 1.

	τ_1^c	τ_2^c	τ_1^u	τ_2^u	o_1	o_2
WCET	5	5	1	5	-	-
Deadline	40	50	-	-	-	-
Period	40	50	-	-	-	-
Valid interval length	-	-	-	-	16	100
LSL	-	-	-	-	37	300

can get their deadlines and all control transactions satisfy Equation (3).

It is worth noting that we need to calculate WCRT of τ_i^c for each value during $[C_i^u, \mathcal{V}_i/2]$ in the worst case. Since $k = \max\{\frac{\mathcal{V}_i}{2} - \max(C_i^u, D_{i-1}^u)\}_{i=1}^n$ and computing WCRT of a periodic task set takes at most $O(nTL^2)$ time, calculating WCRT a periodic task set for satisfying Equation (3) takes at most $O(n \times kTL^2)$. Thus, the overall complexity of Algorithm 2 is $O(n \times (k(n + TL^2) + L))$.

Clearly, DPR-SLG improves the acceptance ratio of minD*-SLG by reassigning a larger deadline to τ_i^u if Equation (3) is not met by τ_i^c after executing minD*. Example 1 shows this advantage of DPR-SLG.

Example 1: Consider a RTDBS with a hybrid transaction set $\mathcal{T} = \{\tau_1^c, \tau_2^c, \tau_1^u, \tau_2^u\}$ and a data object set $\mathcal{O} = \{o_1, o_2\}$ (as shown in Table 2). When executing minD-SLG on $\mathcal{T}$, minD*-SLG first initializes d_1^u and d_2^u to $16/2 = 8$ and $100/2 = 50$, respectively. Then, the periods of τ_1^u and τ_2^u can be obtained by $T_1^u = D_1^u = 8$ and $T_2^u = D_2^u = 50$. Next, QPA is used to test the schedulability of $\mathcal{T}$. Clearly, the result is "schedulable". So, minD* method is executed to assign the deadlines and periods for τ_1^u and τ_2^u, and the result is $D_1^u = 1$, $T_1^u = 15$, $D_2^u = 6$ and $T_2^u = 94$. Since the WCRT of current τ_1^c is no smaller than 17, we have $R_1^c + 2T_1^u \geq 17 + 2 * 15 \geq 47 > 37 > \theta_1$, which means τ_1^c does not satisfy Equation (3) and a "Fail" result will be returned by minD*-SLG. However, when executing DPR-SLG on $\mathcal{T}$, DPR-SLG calculates the minimum increment m by Equation (5). Since the result is $m = 4$, DPR-SLG resets D_1^u to $D_1^u = D_1^u + m + 1 = 1 + 4 + 1 = 6$ and recalculate the WCRT of τ_1^c. Note that, $R_1^c = 17$ can be obtained by most of existing RTA methods (such as Offset-based RTA [39] and Guan's RTA [40]). Since $R_1^c + 2T_1^u = 17 + 2(16 - 6) = 37 = \theta_1$, DPR-SLG sets D_1^u to 6 and recalculates the WCRT of the next control transaction (τ_2^c). It can be obtained that $R_2^c = 17$ and $R_2^c + 2T_2^u = 17 + 2 * (100 - 6) = 205 < 300 < \theta_2$. Therefore, we have that τ_2^c satisfies Equation (3). Since both τ_1^c and τ_2^c satisfy Equation (3), DPR-SLG schedules transactions by EDF and returns the final schedule $\mathcal{T}$.*

V. SIMULATION RESULTS

minD* [36] is an existing deadline and period assignment method for the DPC problem. Since the QoS of control transactions cannot be guaranteed, minD* cannot be directly used for the CS-SLG problem. In our experiments, the performance of minD* is used as the baseline, which is an upper bound of the acceptance ratio. For obtaining the acceptance ratios of minD*, we check the QoS of control transactions on each control transaction absolute deadline in the hyper-period. It should be pointed out that minD* indeed can not be used for the CS-SLG problem since the hyper-period may be too long if the size of the transaction set is large and the periods of transactions are coprime. In this paper, we evaluate the performances of minD*, minD*-SLG and DPR-SLG in terms of execution time, processor workload and acceptance ratio.

TABLE 3. Parameter settings in experiments.

Update transactions		Control transactions		Data objects	
Parameter	Range	Parameter	Range	Parameter	Range
C_i^u	$[1, 15]$	C_i^c	$[1, 15]$	$\mathcal{V}_i$	$[30, 15000]$
-	-	D_i^c	$[20, 5000]$	θ_i	$(\mathcal{V}_i + C_i^c, 35000]$
-	-	T_i^c	$[40, 10000]$	-	-

A. EXPERIMENTAL SETTING

Table 3 shows the parameters in experiments. Note here we use similar baseline values for the parameters as in [12], to keep consistency with previous work. In order to get a uniform distributed transaction utilizations, we use the UUniFast algorithm [41] to generate the transaction utilizations. Bini and Buttazzo [41] showed that the UUniFast algorithm can efficiently generate transaction utilizations with uniform distributions and with $O(n)$ complexity. As shown in Table 3, the WCET of each update transaction satisfies the uniform distribution in [1, 15], and the WCET, deadline and period of each control transaction satisfy the uniform distribution in [1, 15], [20, 5000] and [40, 10000], respectively. The valid interval length of each data object o_i, $\mathcal{V}_i$, satisfies the uniform distribution in [30, 15000], and the LSL of each sample of x_i, θ_i, satisfies the uniform distribution in $(\mathcal{V}_i + C_i^c, 35000]$.

A large number of transaction sets are conducted to evaluate the performances of minD*, minD*-SLG and DPR-SLG. The number of transactions, N, the total utilization of the initial transaction set (in which $D_i^u = T_i^u = \mathcal{V}_i/2$ for each update transaction τ_i^u), U_{total}, and the density factor of update transaction set, $\sum_{i=1}^{n} \frac{C_i^u}{\mathcal{V}_i}$, are three variables in our experiments.

B. EXPERIMENTS UNDER DIFFERENT N

In this set of experiments, the performances of the three methods are tested under different N. An empty initial transaction set is generated at the first step. Then, transactions are conducted and inserted into the initial transaction set one by one until the total utilization of the initial transaction set reaches or exceeds 0.5 (i.e., $U_{total} \geq 0.5$). Next, we change the WCET of the finally inserted transaction to guarantee $U_{total} = 0.5$ if $U_{total} > 0.5$. After obtaining a transaction set with $U_{total} = 0.5$, we record the number of transactions and test the performances of the three methods. A large number of transaction sets are conducted to guarantee that at least 1000 transaction sets can be obtained at each N (in $\{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$). A larger N

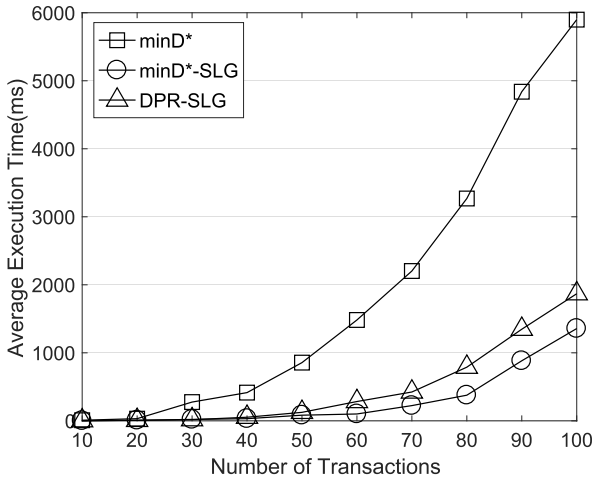


FIGURE 3. Average execution time under $U_{total} = 0.5$.

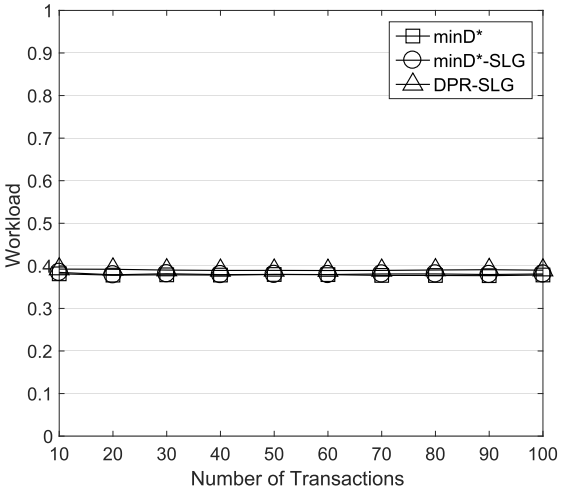


FIGURE 4. Workload under $U_{total} = 0.5$.

implies the transaction set with larger scale, the hyper-period of the transaction set is longer, more update transactions need to calculate deadlines and more control transactions in the transaction set need to be judged whether the QoS of them are guaranteed. All the above results directly increase the execution times of the three algorithms and reduce the acceptance ratios of the three algorithms. The results are shown in Figures 3, 4 and 5.

Comparison on Average Execution Times: As shown in Figure 3, minD*-SLG and DPR-SLG are more efficient than minD* since minD* checks the QoS of control transactions at each control transaction absolute deadline in the hyper-period. The average execution times of minD*-SLG and DPR-SLG are only 23% and 32% of that of minD* when $N = 100$. This is due to minD* needs to judge whether the WCRTs of control transactions at each absolute deadline within the hyper-period are no longer than the LSL of the external objects they assess. In addition, consistent with our expectation, the average execution time of DPR-SLG is longer than that of minD*-SLG. This is because DPR-SLG reassigns

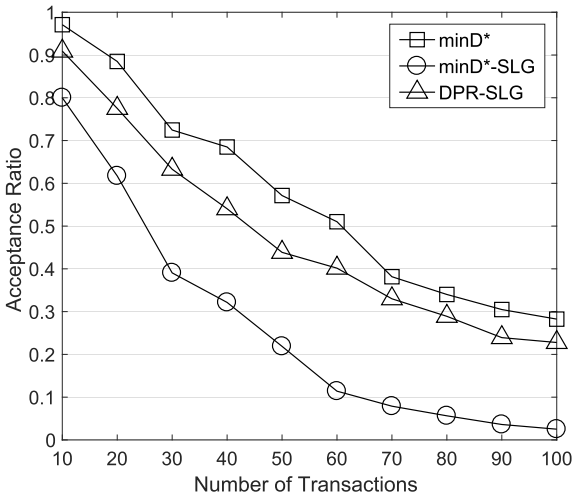


FIGURE 5. Acceptance ratio under $U_{total} = 0.5$.

the deadlines and the periods for update transactions when transaction set can not guarantee the QoS of control transactions after executing minD*. Moreover, the gap between the average execution times (of both minD*-SLG and DPR-SLG) increases with the growth of N and minD*-SLG can reduce the average execution time of DPR-SLG by about 27.4% at $N = 100$. The reason why the gap between the average execution times (of minD*-SLG and DPR-SLG) increases lies in that the WCRTs of transactions increase with the growth of N and DPR-SLG needs to calculate deadlines and periods for more update transactions under a larger N.

Comparison of Processor Workloads: The workload performances of the above three methods are shown in Figure 4. As can be seen, the workload performances of the three algorithms do not change with the growth of N when U_{total} remains unchanged. This indicates that the workload performances of the three algorithms are related to U_{total}. Moreover, the workload of minD* is equal to that of minD*-SLG due to the same deadline and period assignment method used in minD* and minD*-SLG. Another important observation from Figure 4 is that the workload performance of DPR-SLG is consistently slightly higher than that of both minD* and minD*-SLG. The main reason for this is that, compared with minD* and minD*-SLG, DPR-SLG decreases the periods for update transactions when transaction set can not guarantee the QoS of control transactions after executing minD*. This process stops until the transactions set is found to be EDF-schedulable and the QoS of control transactions is guaranteed with those new periods. This indicates that DPR-SLG sacrifices its workload performance to improve its acceptance ratio. In fact, both approaches can derive the near-optimal solution, from the perspective of utilization.

Comparison on Acceptance Ratios: As shown in Figure 5, we can get that the acceptance ratios of the three methods decrease with the growth of N. This is because more control transactions need to be judged whether the QoS of them are guaranteed under a larger N, which increases the possibility that the QoS of control transactions cannot

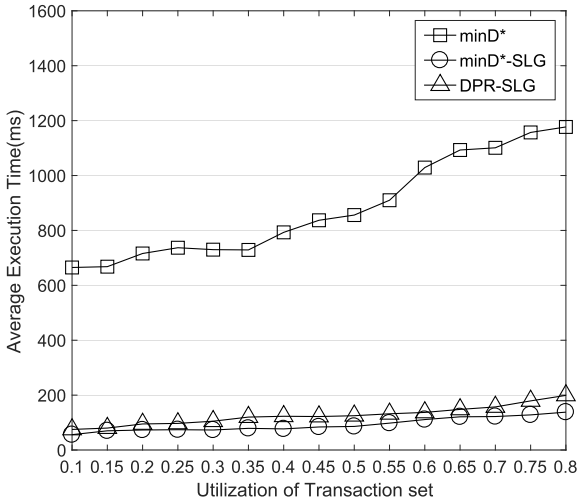


FIGURE 6. Average execution time under $N = 50$.

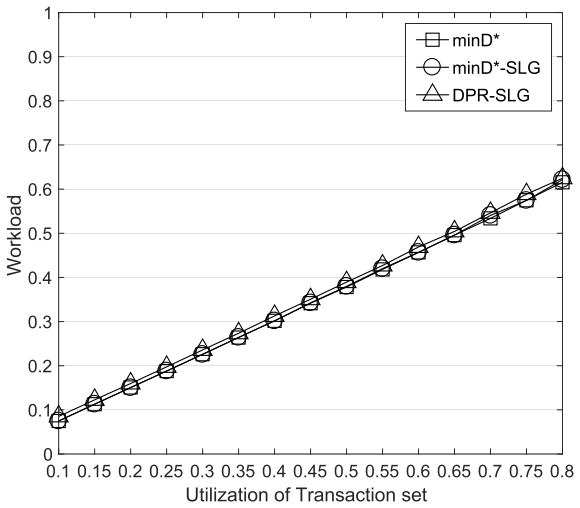


FIGURE 7. Workload under $N = 50$.

be guaranteed. Moreover, the acceptance ratio of DPR-SLG is higher than that of minD*-SLG. This is because DPR-SLG improves the acceptance ratio by increasing the deadlines of update transactions when transaction set can not guarantee the QoS of control transactions after executing minD*. In addition, since some sufficient but unnecessary conditions (for guaranteeing the QoS of control transactions) are used in minD*-SLG and DPR-SLG but an enumeration-based QoS testing method is used in minD*, minD* has the best performance among the three methods.

C. EXPERIMENTS UNDER DIFFERENT U_{TOTAL}

In this set of experiments, we set N to 50 and test the performances of the three methods under different U_{total}s. We group the test results according to the total utilizations of the initial transaction sets and take the average performance as the final results. A large number of transaction sets are conducted to guarantee that at lest 1000 transaction sets can be obtained at each U_{total} (in $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$). A larger U_{total} means that the transactions in the transaction set have higher execution frequency or longer execution time. Thus, the total utilization of the final derived transaction set maybe higher. Figures 6, 7 and 8 show the experimental results, in which a utilization point U_{total} represents the total utilizations in $[U_{total} - 0.01, U_{total} + 0.01]$.

Comparison of Average Execution Times: As shown in Figure 6, DPR-SLG always has the least average execution time among the three methods and the average execution time of minD* is obviously longer than that of DPR-SLG and minD*-SLG. Moreover, both the gap between DPR-SLG and minD* and the gap between minD*-SLG and minD* increase with the growth of U_{total} since the length of the hyper-period increases with the increasing of U_{total}. At $U_{total} = 0.8$, the average execution times of DPR-SLG and minD*-SLG are only 17% and 12% of that of minD*, respectively.

Comparison of Processor Workloads: Figure 7 shows the workload performances of the three methods with different utilizations under $N = 50$. The workload of DPR-SLG is slightly higher than that of minD* and minD*-SLG. The workloads of the three methods increase with the growth of U_{total} due to the increasing of the WCETs. All methods can reduce processor load by about 25%.

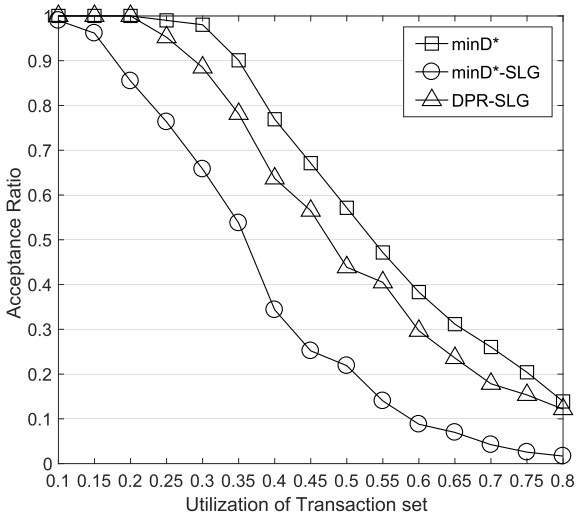


FIGURE 8. Acceptance ratio under $N = 50$.

Comparison of Acceptance Ratios: As shown in Figure 8, the acceptance ratios of the three methods decrease with the growth of U_{total} due to the increasing of the WCETs. Moreover, the gap between minD*-SLG and DPR-SLG increases with the growth of U_{total}. This is because that the WCETs of transactions increase with the growth of U_{total}, which causes the increasing of the WCRTs of control transactions. Since DPR-SLC reassign deadlines for update transactions if the QoS of control transactions cannot be guaranteed, the advantage of DPR-SLG becomes more obvious with the growth of U_{total}.

D. EXPERIMENTS UNDER DIFFERENT DENSITY FACTOR

In this set of experiments, we compare the performances of the three methods under different density factors. We define

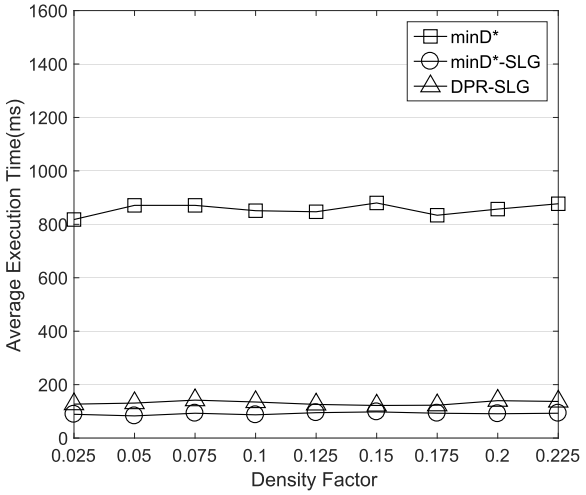


FIGURE 9. Average execution time under $N = 50$ and $U_{total} = 0.5$.

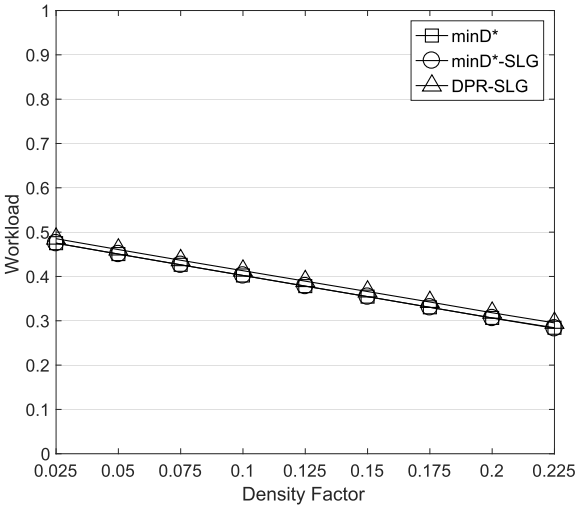


FIGURE 10. Workload under $N = 50$ and $U_{total} = 0.5$.

the density factor for a set of update transactions $\mathcal{T}^u$ as $\sum_{i=1}^{n} \frac{C_i^u}{\mathcal{V}_i}$. At the beginning, we conduct an empty initial transaction set. Then, we generate transaction one by one and insert it into the initial transaction set until U_{total} reaches or exceeds 0.5. Next, we change the WCET of the finally inserted transaction to guarantee $U_{total} = 0.5$ if $U_{total} > 0.5$. After obtaining a transaction set with $U_{total} = 0.5$, we record the transaction set if and only if $N = 50$. A large number of transaction sets are conducted to guarantee that at least 10000 transaction sets can be obtained at $N = 50$, in which at least 100 transaction sets can be obtained at each density factor (in $\{0.025, 0.05, 0.075, 0.1, 0.125, 0.15, 0.175, 0.2, 0.225\}$). Every density factor point represents the density factor in $[\sum_{i=1}^{n} \frac{C_i^u}{\mathcal{V}_i} - 0.01, \sum_{i=1}^{n} \frac{C_i^u}{\mathcal{V}_i} + 0.01]$. A larger density factor implies the update workload is heavier and more workload of update transactions can be reduced by the three methods. The results are shown in Figures 9, 10 and 11.

Comparison of Average Execution Times: Figure 9 depicts the average execution time performances of minD*, minD*-SLG and DPR-SLG with different density factors under $N = 50$ and $U_{total} = 0.5$. As shown in Figure 9, the average execution times of the three methods remain unchanged with the increasing of the density factor. Compared with the other two methods, minD*-SLG has a shorter average execution time. This is due to minD* needs to check the QoS of control transactions at each control transaction absolute deadline in the hyper-period and DPR-SLG needs to reassign the deadlines and the periods for update transactions when the QoS of control transactions cannot be guaranteed after executing minD*.

Comparison of Processor Workloads: Figure 10 compares the workload performances among the three algorithms with different density factors under $N = 50$ and $U_{total} = 0.5$. As shown in the figure, the workloads of the three methods decrease with the growth of the density factor since the total utilization of control transactions decrease with

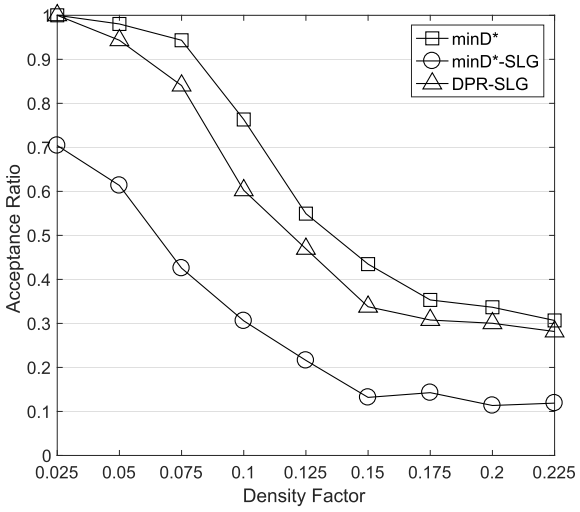


FIGURE 11. Acceptance ratio under $N = 50$ and $U_{total} = 0.5$.

the increasing of the density factor and the total utilization of update transactions can be adjusted by executing the deadline and period assignment method. Moreover, minD* and minD*-SLG have a better performance than DPR-SLG.

Comparison of Acceptance Ratios: Figure 11 shows the acceptance ratios of the three methods with different density factors under $N = 50$ and $U_{total} = 0.5$. It can be viewed that the acceptance ratios of the three methods decrease with growth of the density factor and the acceptance ratio of DPR-SLG is always larger than 90% of that of minD*. The main reason as follows: 1) the WCETs of transactions increase with the growth of density factor, which causes the increasing of the WCRTs of control transactions; 2) DPR-SLG increases the deadlines of update transactions when transaction set cannot guarantee the QoS of control transactions after executing minD*.

E. EXPERIMENT IN LARGE-SCALE TRANSACTION SET

The previous three groups of experiments show the influence of three different parameters on the average execution times,

TABLE 4. Average execution time under $U = 0.5$.

Parameters	Values				
N	200	300	400	500	600
minD*	3.76×10^4	5.01×10^5	9.39×10^5	4.31×10^6	1.43×10^7
minD*-SLG	5.91×10^3	1.46×10^4	5.33×10^4	2.68×10^5	4.07×10^5
DPR-SLG	7.58×10^3	3.08×10^4	7.61×10^4	2.99×10^5	4.54×10^5

TABLE 5. Average length of hyper-period under $U = 0.5$.

Parameters	Values				
N	200	300	400	500	600
Hyper-period	2.06×10^6	8.01×10^6	2.95×10^7	4.76×10^7	1.19×10^8

workloads and acceptance ratios of the three methods. In order to show the practicality of the proposed methods in the large-scale transaction set, we conducted additional experiment by generating a large number of large-scale transaction sets to ensure that at least 100 large-scale transaction sets can be obtained at each N (in $\{200, 300, 400, 500, 600\}$) under $U_{total} = 0.5$.

The average execution time performances of minD*, minD*-SLG and DPR-SLG in large-scale transaction set are shown in Table 4. Consistent with our expectation, the average execution times of both minD*-SLG and DPR-SLG are consistently far less than that of minD*, especially when the N is larger. The average execution times of our proposed methods are less than 10 minutes while the average execution time of minD* is about 4 hours when $N = 600$ and $U_{total} = 0.5$. This is due to the hyper-period becomes much longer when N is very large and minD* needs to make a judgment at every absolute deadline in the hyper-period. The average length of hyper-period of transaction set with different N under $U_{total} = 0.5$ are shown in Table 5. It can be viewed that the hyper-period increases exponentially with the growth of N. In particular, the hyper-period can reach about 1.19×10^8 when $N = 600$ and $U_{total} = 0.5$. Our results show that the effectiveness of the minD*-SLG and DPR-SLG in average execution time over minD* in the large-scale transaction set.

F. SUMMARY OF EXPERIMENT RESULT

It is demonstrated in the experiment results that both minD*-SLG and DPR-SLG give a better performance in terms of average execution time compared with minD*, especially when the scale of transaction set is large. Thus, our proposed methods are effective for the transaction sets with some larger scales. In addition, the workload performance of minD*-SLG is the same as that of minD* and the workload performance of DPR-SLG is always slightly higher than that of both minD* and minD*-SLG. It is worth notig that the gap among them is negligible. Our experiment results show that all methods can reduce the processor workload by about 25%. In general, all methods can effectively reduce the processor workload. Furthermore, DPR-SLG consistently outperforms minD*-SLG in terms of acceptance ratio but slightly lower than that of minD*. DPR-SLG improves the acceptance ratio of minD*-SLG at the price of execution efficiency. The improvement in acceptance ratio is obvious and the efficiencies of both the proposed two methods are acceptable.

VI. CONCLUSION AND FUTURE WORK

In this paper, we define a new problem (CS-SLG) of how to schedule a hybrid transaction set under the guarantee on QoS of control transactions. The service lives of data objects are considered in the CS-SLG problem and control transactions are required to finish their execution before the data objects they get exceed the longest service lives. Two effective methods, minD*-SLG and DPR-SLG, are proposed for resolving the CS-SLG problem. The experimental results show that our proposed methods are effective for the transaction sets with some larger scales and DPR-SLG can improve the acceptance ratio of minD*-SLG at the price of execution efficiency.

For future work, we consider to study the CS-SLG problem in multiprocessor environments. Moreover, the research model of this work limits the number of the transactions accessing or updating each data object. Extending the model to the environments in which a data objects can be accessed and updated by more than one transaction is an another direction to be studied.

REFERENCES

[1] M. Canizo, A. Conde, S. Charramendieta, R. Miñón, R. G. Cid-Fuentes, and E. Onieva, "Implementation of a large-scale platform for cyber-physical system real-time monitoring," *IEEE Access*, vol. 7, pp. 52455–52466, 2019.

[2] T. Gustafsson and J. Hansson, "Data management in real-time systems: A case of on-demand updates in vehicle control systems," in *Proc. RTAS 10th IEEE Real-Time Embedded Technol. Appl. Symp.*, May 2004, pp. 182–191.

[3] X. Shi, Y. Shen, Y. Wang, and L. Bai, "Differential-clustering compression algorithm for real-time aerospace telemetry data," *IEEE Access*, vol. 6, pp. 57425–57433, 2018.

[4] J. Ko, C. Lu, M. B. Srivastava, J. A. Stankovic, A. Terzis, and M. Welsh, "Wireless sensor networks for healthcare," *Proc. IEEE*, vol. 98, no. 11, pp. 1947–1960, Nov. 2010.

[5] S. Han, A. K. Mok, J. Meng, Y. H. Wei, P. C. Huang, Q. Leng, X. Zhu, L. Sentis, K. S. Kim, and R. Miikkulainen, "Architecture of a cyberphysical avatar," in *Proc. ACM/IEEE Int. Conf. Cyber-Phys. Syst.*, Philadelphia, PA, USA, Apr. 2013, pp. 189–198.

[6] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. ACM (JACM)*, vol. 20, no. 1, pp. 46–61, Jan. 1973.

[7] J. Y.-T. Leung and J. Whitehead, "On the complexity of fixed-priority scheduling of periodic, real-time tasks," *Perform. Eval.*, vol. 2, no. 4, pp. 237–250, Dec. 1982.

[8] M. L. Dertouzos, "Control robotics:The Procedural Control of Physical Processes," in *Proc. IFIP Congr.*, 1974, pp. 807–813.

[9] M. Xiong, Q. Wang, and K. Ramamritham, "On earliest deadline first scheduling for temporal consistency maintenance," *Real-Time Syst.*, vol. 40, no. 2, pp. 208–237, 2008.

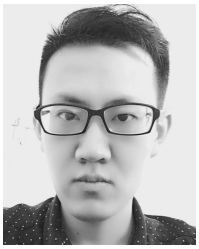
[10] M. Xiong, Q. Wang, and K. Ramamritham, "Scheduling to minimize staleness and stretch in real-time data warehouses," *Theory Comput. Syst.*, vol. 49, no. 4, pp. 757–780, Nov. 2011.

[11] L. Golab, T. Johnson, and V. Shkapenyuk, "Scheduling updates in a real-time stream warehouse," in *Proc. IEEE 25th Int. Conf. Data Eng.*, Mar. 2009, pp. 1207–1210.

[12] S. Han, K.-Y. Lam, J. Wang, K. Ramamritham, and A. K. Mok, "On co-scheduling of update and control transactions in real-time sensing and control systems: Algorithms, analysis, and performance," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 10, pp. 2325–2342, Oct. 2013.

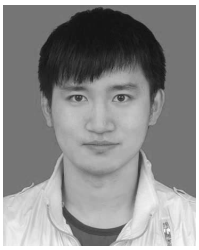
[13] S.-J. Ho, T.-W. Kuo, and A. K. Mok, "Similarity-based load adjustment for real-time data-intensive applications," in *Proc. Real-Time Syst. Symp.*, Dec. 1997, pp. 144–157.

[14] M. Xiong, S. Han, and K.-Y. Lam, "A deferrable scheduling algorithm for real-time transactions maintaining data freshness," in *Proc. IEEE Real-Time Syst. Symp.*, Miami, FL, USA, Dec. 2005, pp. 27–37.
[15] M. Xiong and K. Ramamritham, "Deriving deadlines and periods for real-time update transactions," *IEEE Trans. Comput.*, vol. 53, no. 5, pp. 567–583, May 2004.
[16] R. Huang, H. Liang, J. Chen, P. Zhao, and M. Du, "Lidar based dynamic obstacle detection, tracking and recognition method for driverless cars," *Robot*, vol. 38, no. 4, pp. 437–443, Jun. 2016.
[17] J. Bi, H. Yuan, W. Tan, M. Zhou, Y. Fan, J. Zhang, and J. Li, "Application-aware dynamic fine-grained resource provisioning in a virtualized cloud data center," *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 2, pp. 1172–1184, Apr. 2017.
[18] H. Yuan, J. Bi, W. Tan, M. Zhou, B. Hu Li, and J. Li, "TTSA: An effective scheduling approach for delay bounded tasks in hybrid clouds," *IEEE Trans. Cybern.*, vol. 47, no. 11, pp. 3658–3668, Nov. 2017.
[19] H. Yuan, J. Bi, W. Tan, and B. Hu Li, "Temporal task scheduling with constrained service delay for profit maximization in hybrid clouds," *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 1, pp. 337–348, Jan. 2017.
[20] H. Yuan, J. Bi, W. Tan, and B. Hu Li, "CAWSAC: Cost-aware workload scheduling and admission control for distributed cloud data centers," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 2, pp. 976–985, Apr. 2016.
[21] H. Yuan, J. Bi, M. Zhou, and A. C. Ammari, "Time-aware multi-application task scheduling with guaranteed delay constraints in green data center," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 3, pp. 1138–1151, Jul. 2018.
[22] Gerber, Hong, and Saksena, "Guaranteeing end-to-end timing constraints by calibrating intermediate processes," in *Proc. Real-Time Syst. Symp.*, Dec. 1994, pp. 192–203.
[23] A. Kumar Jha, M. Xiong, and K. Ramamritham, "Mutual consistency in real-time databases," in *Proc. 27th IEEE Int. Real-Time Syst. Symp. (RTSS)*, Dec. 2006, pp. 335–343.
[24] K.-D. Kang, S. H. Son, J. A. Stankovic, and T. F. Abdelzaher, "A QoS-sensitive approach for timeliness and freshness guarantees in real-time databases," in *Proc. 14th Euromicro Conf. Real-Time Syst. Euromicro RTS*, Jun. 2002, pp. 203–212.
[25] Y. Kim and S. Son, "Predictability and consistency in real-time database systems," *IEEE Real-Time Syst. Newslett.*, vol. 5, nos. 2–3, pp. 42–45, Mar. 1993.
[26] K.-Y. Lam, M. Xiong, B. Yu Liang, and Y. Guo, "Statistical quality of service guarantee for temporal consistency of real-time data objects," in *Proc. 25th IEEE Int. Real-Time Syst. Symp.*, Dec. 2004, pp. 276–285.
[27] X. Song and J. W. S. Liu, "Maintaining temporal consistency: Pessimistic vs. Optimistic concurrency control," *IEEE Trans. Knowl. Data Eng.*, vol. 7, no. 5, pp. 786–796, Oct. 1995.
[28] T.-W. Kuo and A. K. Mok, "SSP: A semantics-based protocol for real-time data access," in *Proc. Real-Time Syst. Symp.*, Dec. 1993, pp. 76–86.
[29] M. Xiong, K. Ramamritham, J. A. Stankovic, D. Towsley, and R. Sivasankaran, "Scheduling transactions with temporal constraints: Exploiting data semantics," *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 5, pp. 1155–1166, Oct. 2002.
[30] M. Xiong, S. Han, K.-Y. Lam, and D. Chen, "Deferrable scheduling for maintaining real-time data freshness: Algorithms, analysis, and results," *IEEE Trans. Comput.*, vol. 57, no. 7, pp. 952–964, Jul. 2008.
[31] S. Han, D. Chen, M. Xiong, K.-Y. Lam, A. K. Mok, and K. Ramamritham, "Schedulability analysis of deferrable scheduling algorithms for maintaining real-time data freshness," *IEEE Trans. Knowl. Data Eng.*, vol. 63, no. 4, pp. 979–994, Apr. 2014.
[32] S. Han, D. Chen, M. Xiong, and A. K. Mok, "Online scheduling switch for maintaining data freshness in flexible real-time systems," in *Proc. 30th IEEE Real-Time Syst. Symp.*, Dec. 2009, pp. 115–124.
[33] F. Zhu, J. Li, and G. Li, "An efficient deadline and period assignment scheme for maintaining temporal consistency under EDF," in *Proc. Int. Conf. Hum.-Centric Comput. Embedded Multimedia Comput.*, Aug. 2011, pp. 351–364.
[34] S. Han, K.-Y. Lam, J. Wang, S. H. Son, and A. K. Mok, "Adaptive co-scheduling for periodic application and update transactions in real-time database systems," *J. Syst. Softw.*, vol. 85, no. 8, pp. 1729–1743, Aug. 2012.
[35] J.-T. Wang, K.-Y. Lam, S. Han, S. H. Son, and A. K. Mok, "An effective fixed priority co-scheduling algorithm for periodic update and application transactions," *Computing*, vol. 95, nos. 10–11, pp. 993–1018, Oct. 2013.
[36] G. Li, C. Deng, J. Li, Q. Zhou, and W. Wei, "Deadline and period assignment for update transactions in co-scheduling environment," *IEEE Trans. Comput.*, vol. 66, no. 7, pp. 1119–1131, Jul. 2017.
[37] K. Ramamritham, "Real-time databases," *Distrib. Parallel Databases*, vol. 1, no. 2, pp. 199–226, Apr. 1993.
[38] F. Zhang and A. Burns, "Schedulability analysis for real-time systems with EDF scheduling," *IEEE Trans. Comput.*, vol. 58, no. 9, pp. 1250–1258, Sep. 2009.
[39] J. C. Palencia and M. G. Harbour, "Offset-based response time analysis of distributed systems scheduled under EDF," in *Proc. 15th Euromicro Conf. Real-Time Syst.*, Jul. 2003, pp. 3–12.
[40] N. Guan and W. Yi, "General and efficient response time analysis for EDF scheduling," in *Proc. Conf. Design, Autom. Test Eur.*, Dresden, Germany, Mar. 2014, pp. 255–260.
[41] E. Bini and G. C. Buttazzo, "Measuring the performance of schedulability tests," *Real-Time Syst.*, vol. 30, nos. 1–2, pp. 129–154, May 2005.



CHENGGANG DENG received the Ph.D. degree in computer science from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2017. He holds a postdoctoral position with the School of Computer Science and Technology, HUST. His research interests include real-time database and real-time scheduling.

GUOHUI LI received the Ph.D. degree in computer science from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 1999. He was a Full Professor, in 2004. He is currently the Dean of the School of Software Engineering, HUST. His main research interests include real-time systems, mobile computing, and advanced data management.



QUAN ZHOU received the Ph.D. degree in computer science from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2015. He held a postdoctoral position with the School of Computer Science and Technology, HUST, where he is currently an Assistant Professor. His research interests include real-time scheduling and mobile computing.

JIANJUN LI received the Ph.D. degree in computer science from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2012. He was a Senior Research Associate with the Department of Computer Science, City University of Hong Kong, Hong Kong. He is currently an Associate Professor with the School of Computer Science and Technology, HUST. His research interests include real-time systems and advanced data management.