

2017

A Markov-Based Update Policy for Constantly Changing Database Systems

Wei Zong

Xian Jiaotong University


Feng Wu

Xidian University

Zhengrui Jiang

Iowa State University, zjiang@iastate.edu

Follow this and additional works at: http://lib.dr.iastate.edu/scm_pubs

 Part of the [Management Information Systems Commons](#), [Marketing Commons](#), [Operations and Supply Chain Management Commons](#), and the [Technology and Innovation Commons](#)

The complete bibliographic information for this item can be found at http://lib.dr.iastate.edu/scm_pubs/31. For information on how to cite this item, please visit <http://lib.dr.iastate.edu/howtocite.html>.

This Article is brought to you for free and open access by the Supply Chain and Information Systems at Iowa State University Digital Repository. It has been accepted for inclusion in Supply Chain and Information Systems Publications by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.

A Markov-Based Update Policy for Constantly Changing Database Systems

Abstract

In order to maximize the value of an organization's data assets, it is important to keep data in its databases up-to-date. In the era of big data, however, constantly changing data sources make it a challenging task to assure data timeliness in enterprise systems. For instance, due to the high frequency of purchase transactions, purchase data stored in an enterprise resource planning system can easily become outdated, affecting the accuracy of inventory data and the quality of inventory replenishment decisions. Despite the importance of data timeliness, updating a database as soon as new data arrives is typically not optimal because of high update cost. Therefore, a critical problem in this context is to determine the optimal update policy for database systems. In this study, we develop a Markov decision process model, solved via dynamic programming, to derive the optimal update policy that minimizes the sum of data staleness cost and update cost. Based on real-world enterprise data, we conduct experiments to evaluate the performance of the proposed update policy in relation to benchmark policies analyzed in the prior literature. The experimental results show that the proposed update policy outperforms fixed interval update policies and can lead to significant cost savings.

Keywords

Data timeliness, update policy, data quality, Enterprise Resource Planning, Markov decision process

Disciplines

Management Information Systems | Marketing | Operations and Supply Chain Management | Technology and Innovation

Comments

This is an accepted manuscript published as Zong, W., F. Wu, Z. Jiang. "A Markov-based Update Policy for Constantly Changing Database Systems." IEEE Transactions on Engineering Management, August 2017; 64(3);287-300. [10.1109/TEM.2017.2648516](https://doi.org/10.1109/TEM.2017.2648516) . Posted with permission.

Rights

Personal use of this material is permitted. Permission from IEEE must be obtained for all other users, including reprinting/ republishing this material for advertising or promotional purposes, creating new collective works for resale or redistribution to servers or lists, or reuse of any copyrighted components of this work in other works

A Markov-based Update Policy for Constantly Changing Data in Database Systems

Wei Zong^{a, b}, Feng Wu^b, Zhengrui Jiang^c

wei.zong@stu.xjtu.edu.cn, fengwu@mail.xjtu.edu.cn, zjiang@iastate.edu

^aSchool of Economics and Management, Xidian University, No.2 South Taibai Road, Xi'an 710071, PR China

^bSchool of Management, Xi'an Jiaotong University, 28 Xianning west road, Xi'an 710049, PR China

^cCollege of Business, Iowa State University, 2340 Gerdin Business Building, Ames, Iowa 50011, USA

Abstract—In order to maximize the value of an organization's data assets, it is important to keep data in its databases up-to-date. In the era of big data, however, constantly changing data sources make it a challenging task to assure data timeliness in enterprise systems. For instance, due to high frequency of purchase transactions, purchase data stored in an Enterprise Resource Planning (ERP) system can easily become outdated, affecting the accuracy of inventory data and the quality of inventory replenishment decisions. Despite the importance of data timeliness, updating a database as soon as new data arrives is typically not optimal because of high update cost. Therefore, a critical problem in this context is to determine the optimal update policy for database systems. In this study, we develop a Markov decision process model, solved via dynamic programming, to derive the optimal update policy that minimizes the sum of data staleness cost and update cost. Based on real-world enterprise data, we conduct experiments to evaluate the performance of the proposed update policy in relation to benchmark policies analyzed in the prior literature. The experimental results show that the proposed update policy outperforms fixed interval update policies and can lead to significant cost savings.

Managerial Relevance — In the era of big data, most organizations understand the business value of data, and are collecting an unprecedented amount of data from different sources. For efficient decision support, such data should be kept up-to-date. However, the sheer volume of constantly changing data makes it increasingly difficult to achieve and maintain data timeliness. Therefore, more than ever, an efficient database update policy is critical if organizations want to maximize the value derived from their data assets. This research addresses this problem by developing a Markov decision process model to help decide when and how frequently an organization should update its database system. The aperiodic update policy derived in this study takes into consideration the tradeoffs between data staleness cost and update cost, and outperforms fixed interval policies proposed in the prior literature. As a result, the proposed policy can help maximize the value of organizations' data assets. Moreover, we believe that the data update policy proposed in this research can be adapted to develop cost efficient maintenance policies for a wide range of tangible and intangible assets, and help organizations achieve significant financial savings.

Index Terms — Data timeliness, update policy, data quality, Enterprise Resource Planning, Markov decision process

A Markov-based Update Policy for Constantly Changing Data in Database Systems

I. INTRODUCTION AND MOTIVATION

Keeping data up-to-date in a database system is critical in order to maximize the value of an organization's data assets. However, constantly changing data sources often make it a challenging task to assure data timeliness in enterprise database systems. Because Enterprise Resource Planning (ERP) systems are one of the most widely adopted types of software systems in today's organizations [1, 2], we use ERP systems to illustrate the challenges, the solutions, and the benefits of achieving data timeliness in a modern database systems.

By integrating processes and data from internal functions of an organization and other sources, a successful ERP system can significantly improve the organization's efficiency, productivity, and competitiveness in the marketplace [3, 4]. However, the advantages that an ERP provides for an organization depend on the quality of data it processes. In fact, data quality has been found to be one of the critical factors in the successful operation of ERP systems [5, 6].

In the era of big data, data often arrives from multiple sources at a high velocity, which brings great challenges to data management [7, 8]. Under this circumstance, organizations hope their data can be updated quickly to capture the data changes, so that at the time of application, current and high quality data can be extracted to support effective decision-making. In fact, data timeliness is identified as one of the most important data quality dimensions for database systems [9, 10]. This is because only when the data in an ERP system is kept current, can effective decisions be made. If the data in a database system is out of date, decisions made

based on obsolete data can be incorrect or ineffective, leading to possible financial losses. Take the inventory data in an ERP system as an example, if a new purchase order has been filled, but the new purchase data is not updated in the system in a timely manner, then the company may incur data staleness cost as a result of inefficient inventory replenishment decisions made based on the inaccurate and stale inventory data.

An obvious solution for such data staleness problem is to update the database as soon as a change in the data source occurs, but such a policy is typically inefficient or even infeasible in practice, because it is very costly to keep the data in a database system up to date in real-time. Sometimes the database may be under heavy use to support business operations, hence the update has to be delayed. There are also scenarios where database update often cannot be completed without human intervention. In the latter case, the personnel cost could be much higher than the equipment and computation cost [11]. Furthermore, updating information systems frequently can affect system availability [12]. Therefore, to decide when and how to update data in a database system, we need to carefully construct analytical models to balance the trade-off between staleness cost and update cost.

The remainder of the paper is organized as follows. Section 2 provides a brief introduction to the data timeliness problem and an overview of the main research efforts on data update policies in the prior literature. Section 3 introduces the purchase data update problem and the analytical models. The analysis of the optimal update policy and computation method is provided in Section 4. Section 5 reports on the experiments conducted to evaluate the effectiveness of the proposed update policy in relation to benchmark policies. Finally, Section 6 presents some concluding remarks and discusses possible future research directions.

II. RELATED LITERATURE

This section first provides an introduction to how data timeliness is defined in the extant literature, followed by a review of data update policies proposed in prior studies.

A. Current Research on Data Timeliness

Data timeliness has long been considered an important data quality dimension and needs improvement for effective decision making, especially in the age of big data[13]. Prior research has tried to define data timeliness and proposed various timeless metrics in different applications. Wand and Wang [14] defined timeliness as the delay between a change of a real world state and the resulting modification of the recorded state in an information system. Similarly, Lee and Strong [15] considered timeliness the extent to which the data was sufficiently up-to-date for the task at hand. Ballou and Pazer [16] provided a more general and pragmatic definition and employed currency and volatility to judge whether the data was out of date. Currency is the difference between the time when data changes in the real-world and the time when users use it for a certain task. Volatility refers to the average time interval that data remains valid.

Based on the definitions in the aforementioned studies, we can see that data timeliness is affected by three important time instances, i.e. the time when data changes in the real world, the time when the changes are extracted and recorded in an information system, and the time when the data is delivered to the information users. The entire process and related events affecting data timeliness are depicted in Fig. 1.

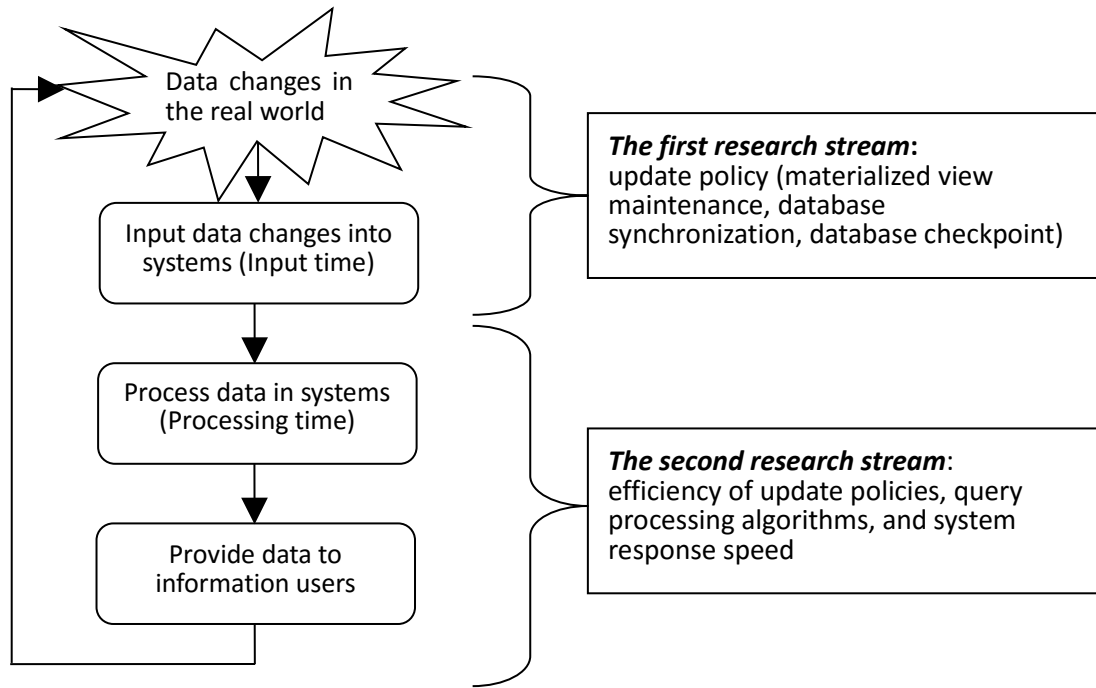


Fig. 1. Events and classification of prior research on data timeliness.

Based on the events affecting data timeliness shown in Fig. 1, the literature in the area of data timeliness can be classified into two research streams. The first research stream mainly focuses on improving data timeliness by quickly extracting and recording the data changes in the real-world into an information system, i.e. the data update policy for databases. The second research stream focuses on improving data timeliness by quickly delivering the data changes or the requested information quickly to information users. The second stream features mainly study on how to improve the efficiency of update policies [10, 17, 18], query processing algorithms [9, 19, 20], and system response speed [21, 22].

In this paper, the research problem we focus on designing optimal data update policies to achieve data timeliness. Data timeliness in this paper refers to the difference between the time when data changes in the real-world and the time when these changes are input into a database system, which corresponds to the first two events in Fig. 1. Therefore, our research belongs to

the first research stream and a more detailed review of the related literature on update policy is provided next.

B. Current Research on Data Update Policy

The process of updating the data contents in information systems to reflect the changes occurring at the data sources is termed as synchronization or materialized view maintenance in some prior studies. Chandy, Browne, Dissly and Uhrig [23] presented models and techniques to determine the optimal time interval for database checkpoints. Srivastava and Rotem [24] focused on the efficiency of data update operation and obtained the optimal update frequency based on a stochastic model by minimizing a linear weighted sum of the query waiting cost and processing cost. However, the data staleness cost was not explicitly considered in their research. Segev and Fang [25] designed and proposed the optimal time-based and query-based data update policies, respectively. The objective of their models was to minimize the currency on user queries rather than minimizing the data staleness cost. According to the different properties of updates and database views, Adelberg, Garcia-Molina and Kao [26] proposed some data update strategies by analyzing various properties of updates and views to balance the transaction deadlines with database currency. Ling and Mi [27] and Dey, Zhang and De [28] discussed a time-based update policy and designed the optimal update frequency by minimizing the total data staleness cost and synchronization cost. Mannino and Walter [29] explored the short-term and long-term influences of data refresh policies on the traditional information system success measures by conducting a field study. However, a common characteristic of these studies is that the optimal data update policies derived are mainly based on a fixed interval. That is to say, the database is updated after a fixed time interval, e.g., one

day/week after a fixed number of user queries have been received by a database, or after a fixed number of new data changes have arrived at a database. To increase the flexibility of data update process, we propose an aperiodic data update policy, illustrated using purchase data update in ERP systems, and test its effectiveness and optimality by comparing it with traditional fixed interval data update policies. Although aperiodic policies have been studied in [11] and [30], the former study focuses on refreshing the knowledge learned from data (e.g., association rules computed from sales data) instead of updating the data itself, and the decision variables, decision models, and cost function considered are also different from ours, leading to different update policies.

III. DESCRIPTION ON RESEARCH PROBLEM AND MODEL

Although the database update policy proposed in this study is applicable in a wide range of data and systems, as explained earlier, for ease of exposition, we use purchase data in an ERP system to motivate the problem and illustrate the model and solutions.

A. Purchase Data Update Process in ERP Systems

Purchase data plays an important role in decision-making in manufacturing companies. It is used not only for studying past purchase patterns, but also for updating inventory status of materials and products. For ease of illustration, we consider one type of information inquiry, i.e. inventory inquiry, to an ERP system. User inquiries of inventory status for materials are frequently posted to the ERP system to check the current inventory level and support inventory replenishment decisions. Due to the continuously arriving purchase data, the purchase and inventory data recorded previously may no longer be valid and accurate when an inventory inquiry arrives. Therefore, the purchase data in the ERP system needs to be updated in a timely

manner to reflect the changes occurring at purchase data sources and provide more accurate data to support inventory replenishment decisions. If the ERP system is not updated with the new purchase data, the inventory inquiry will receive outdated data, which can result in suboptimal decisions and possible financial losses to the company. Conversely, updating the ERP system immediately on the arrival of new purchase data can ensure the timeliness and freshness of the data, but the resulting frequent updates lead to higher update cost. Therefore, a decision regarding whether or not to update the ERP system has to be made when (or just before) an inventory inquiry arrives at the ERP system in a way that minimizes the sum of staleness cost and update cost. The timing of purchase data update decisions can be illustrated in Fig. 2.

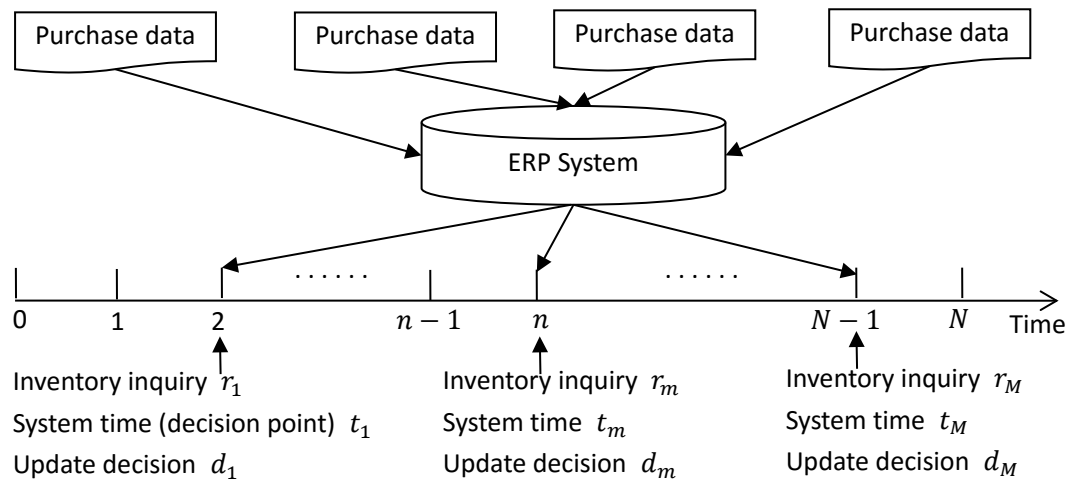


Fig. 2. Timing of purchase data update decisions.

B. Definition and Description on the Research Model

In Fig. 2, the data update decisions are discrete, i.e. the data update decision has to be made at each decision point, and each data update decision is only determined by the current system state instead of the previous system states. This conforms to the characteristics of Markov decision process. Therefore, we model the purchase data update problem in ERP as a Markov

decision process. System state and transition probability are two indispensable parameters in a Markov decision process. Therefore in this section, we start with the definitions and analysis of system state and transition probability, followed by the derivation of system cost and objective function. The key notations used in this paper are listed in TABLE AI in the **APPENDIX**.

C. Analysis of System State and Transition Probability

Many prior studies have assumed that information inquiries and data updates to a database approximately follow a Poisson process [27, 28, 31]. Following the prior literature, we also assume that the arrival of new purchase data to an ERP system is characterized by a Poisson process with an intensity rate of λ_u and that inventory inquiries arriving to an ERP system follow another independent Poisson process with an intensity rate of λ_r . The expected number of inventory inquiries submitted to an ERP system in a predetermined time horizon is denoted by M . The system time t_m refers to the time when the m^{th} inventory inquiry arrives at an ERP system, where $m = 1, 2, \dots, M$. t_m can also be considered the decision points because an update decision d_m needs to be made when the m^{th} inventory inquiry arrives at an ERP system. The update decision d_m can take one of two possible values, 0 and 1. When $d_m = 0$, it means that the ERP system is not updated. When $d_m = 1$, it indicates that the ERP system is updated with the accumulated purchase data incorporated into the system. As expected, the value of d_m is determined by the system state s_m . In this study, we define system state s_m as the quantity of unprocessed purchase records accumulated by time t_m , where a purchase record describes the purchase details for a particular type of material, such as the material name, purchase date, purchase quantity, and purchase price, etc. The decision made at system time

t_m will affect the system state at the next system time t_{m+1} . Due to the different values of d_m , the system state at the next decision point should be examined separately.

If $d_m = 0$, it means that the ERP system is not updated at system time t_m . As a result, the accumulated and unprocessed purchase data at time t_m will be carried over to the next system time t_{m+1} . Conversely, if $d_m = 1$, the ERP system is updated at system time t_m . Then the accumulated purchase data at time t_m will be integrated into the ERP system instead of being carried over to the next system time t_{m+1} . Therefore, the system state s_{m+1} with different update decisions can be represented by:

$$s_{m+1} = \begin{cases} s_m + I_{m,m+1} & d_m = 0, \\ I_{m,m+1} & d_m = 1. \end{cases} \quad (1)$$

where $I_{m,m+1}$ is the quantity of purchase data accumulated from time t_m to t_{m+1} .

The system transition from the current system state s_m to the next system state s_{m+1} is captured by the transition probability $P_{s_m, d_m, s_{m+1}}$. As illustrated in Fig. 2, when the first inventory inquiry arrives at the ERP system at system time t_1 , an update decision d_1 needs to be made under the current system state s_1 . After that, the ERP system stochastically transits to the next state s_2 with a transition probability P_{s_1, d_1, s_2} . At the system state s_2 , a decision d_2 is made and the ERP system stochastically transits to the next state s_3 with a transition probability P_{s_2, d_2, s_3} , and so on. Fig. 3 illustrates the process of system state transition during purchase data update in ERP systems.

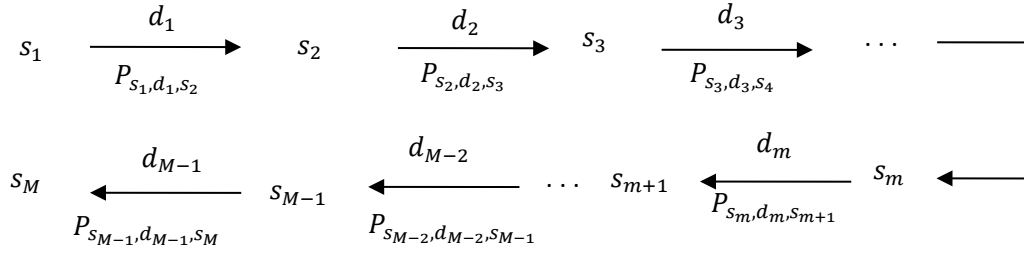


Fig. 3. Process of system state transition.

In the above analysis, we have assumed that the arrival of inventory inquiries follows a Poisson process with an intensity rate of λ_r . Under this assumption, the time interval between two successive inventory inquiries follows the exponential distribution with an intensity rate of λ_r . Therefore, the probability of the data quantity accumulated from time t_m to t_{m+1} , i.e.

$P(I_{m,m+1} = h)$, takes the form,

$$P(I_{m,m+1} = h) = \int_0^\infty P(I_{m,m+1} = h | (t_{m+1} - t_m)) \cdot f(t_{m+1} - t_m) d(t_{m+1} - t_m). \quad (2)$$

where $f(t_{m+1} - t_m)$ is the probability density function of time interval $(t_{m+1} - t_m)$ and $f(t_{m+1} - t_m) = \lambda_r e^{-\lambda_r t}$. Let $(t_{m+1} - t_m) = t$, then Eq. (2) can be rewritten as:

$$\begin{aligned} P(I_{m,m+1} = h) &= \int_0^\infty P(I_{m,m+1} = h | t) \cdot f(t) d(t) = \int_0^\infty \frac{e^{-\lambda_u t} \cdot (\lambda_u t)^h}{h!} \cdot \lambda_r e^{-\lambda_r t} dt \\ &= \frac{(\lambda_u)^h \cdot \lambda_r}{(\lambda_u + \lambda_r)^{h+1}}. \end{aligned} \quad (3)$$

There are two possible values for d_m ($d_m \in \{0,1\}$), therefore the value of $P_{s_m, d_m, s_{m+1}}$ should be discussed separately.

When $d_m = 0$, from Eq. (1) we know that:

$$I_{m,m+1} = s_{m+1} - s_m. \quad (4)$$

Replacing parameter h in Eq. (3) with Eq. (4), we get:

$$P_{s_m, d_m=0, s_{m+1}} = P(I_{m,m+1} = s_{m+1} - s_m) = \frac{(\lambda_u)^{(s_{m+1}-s_m)} \cdot \lambda_r}{(\lambda_u + \lambda_r)^{(s_{m+1}-s_m+1)}}. \quad (5)$$

When $d_m = 1$, according to Eq. (1) we have:

$$I_{m,m+1} = s_{m+1}. \quad (6)$$

In this case, substituting h in Eq. (3) with Eq. (6) yields:

$$P_{s_m, d_m=1, s_{m+1}} = P(I_{m,m+1} = s_{m+1}) = \frac{(\lambda_u)^{s_{m+1}} \cdot \lambda_r}{(\lambda_u + \lambda_r)^{(s_{m+1}+1)}}. \quad (7)$$

D. Analysis of System Cost

Due to the different cases of update decision made at system time t_m , the system cost $c_m(s_m, d_m)$ should also be discussed separately. If $d_m = 0$, i.e., the ERP system is not updated at system time t_m , hence the data obtained to answer the inventory inquiry r_m is stale and the staleness cost is incurred, as represented by $c_s(s_m)$ in Eq. (8) below. On the other hand, if $d_m = 1$, i.e. the ERP system is updated at system time t_m , the staleness cost will be avoided but the update cost is incurred, as represented by c_u in Eq. (8). In line with prior research (e.g., Dey *et al.* 2006), we assume that the update cost is a constant and independent of the amount of newly arriving data [28]. In this way, the system cost $c_m(s_m, d_m)$ at t_m with different update decisions can be defined by Eq. (8):

$$c_m(s_m, d_m) = \begin{cases} c_s(s_m) & d_m = 0, \\ c_u & d_m = 1. \end{cases} \quad (8)$$

Based on the update decision and the corresponding cost at system time t_m , we can formulate the objective function of the purchase data update problem as:

$$C = \min E\left(\sum_{m=1}^M (d_m \cdot c_u + (1 - d_m) \cdot c_s(s_m))\right). \quad (9)$$

For the ease of analysis, Eq. (9) can be rewritten as:

$$C = \min E(c_1(s_1, d_1) + c_2(s_2, d_2) + \dots + c_M(s_M, d_M)). \quad (10)$$

The objective of problem (10) is to minimize the expected total system cost in a time horizon by finding the optimal sequence of update decisions δ^* at all decision points, i.e., $\delta^* = (d_1^*, d_2^*, \dots, d_M^*)$. If we consider one decision point at a time, the optimal update decision

d_m^* at time t_m can be decided by comparing $c_s(s_m)$ with c_u . If $c_s(s_m) \geq c_u$, it is better to update the ERP system and therefore $d_m^* = 1$. If $c_s(s_m) < c_u$, the smaller data staleness cost indicates that there's no need to update the ERP system, therefore $d_m^* = 0$. However, with all decision points considered, the above mentioned simple method is generally not optimal. Therefore, we need to develop an efficient method to obtain the optimal update sequence for the entire time horizon.

E. Analysis of Staleness Cost Function

Staleness cost $c_s(s_m)$ reflects the effect of accumulated unprocessed purchase data on future purchase decisions. In an ERP system, if an inventory inquiry is answered by the stale data, the purchase decisions made based on those outdated inventory data will likely be inefficient and inaccurate, leading to the unnecessary *purchase cost* c_p and *inventory cost* c_I . Therefore, in the present research the staleness cost $c_s(s_m)$ is defined as:

$$c_s(s_m) = (c_p + c_I) \cdot F(s_m), \quad (11)$$

where purchase cost c_p and inventory cost c_I are considered constant. $F(s_m)$, which we refer to as the *staleness severity function*, represents the severity of inefficiency or financial loss as a result of data staleness in the ERP systems and is a function of system state s_m . Note that similar concept of staleness severity has been proposed in prior research (e.g., [27,28]). Recall that s_m represents the accumulated quantity of unprocessed purchase data at time t_m , therefore, we can see from Eq. (11) that the staleness cost is dependent on the quantity of accumulated unprocessed purchase data at t_m . The larger the amount of unprocessed purchase data at t_m , the more likely the focal company will make inefficient inventory replenishment decisions, thus resulting in a higher staleness cost. Therefore, $c_s(s_m)$ is a monotone increasing

function in system state s_m . Furthermore, we define $F(s_m) = 0$ when $s_m = 0$. Then we have $c_s(s_m) = 0$, indicating that if the quantity of unprocessed purchase data accumulated by time t_m is zero, i.e., the purchase data in ERP system is up to date at t_m , then there will be no stale data and the staleness cost is zero.

By analyzing the characteristics of Eq. (10), we can find that the staleness severity function $F(s_m)$ plays a critical role in determining the value of the objective function. Theorem 1 describes the relationship between the optimal expected total system cost and the staleness severity function.

THEOREM 1. For a same system state s_m , let C_1 and C_2 denote the optimal expected total system cost corresponding to staleness severity $F_1(s_m)$ and $F_2(s_m)$. If $F_1(s_m) \geq F_2(s_m)$, then $C_1 \geq C_2$, where $s_m \in S$ and $m = 1, 2, \dots, M$. (All proofs are provided in the **APPENDIX**.)

As explained earlier, $F(s_m)$ represents the severity of inefficiency or financial loss as a result of data staleness. Based on Theorem 1, for the same system state s_m , a higher value of $F(s_m)$ indicates that the cost of stale data is higher. This shows from another perspective that the company with a higher staleness severity function is more sensitive to stale data, and therefore may put more emphasis on maintaining data timeliness for decisions making.

IV. ALGORITHM FOR DERIVING THE OPTIMAL UPDATE POLICY

In order to obtain the minimal expected total cost as specified in the objective function of problem (10), we need to examine all system state s_m at each decision point d_m for $m = 1, 2, \dots, M$. This seems impossible because the number of possible values of system state s_m is infinite due to the stochastic nature of the accumulated purchase data at t_m . Fortunately, based

on the above analysis we can see that, each data update decision at each decision point is only determined by the current system state instead of the previous system states. Therefore, the entire update decision problem can be formulated as a Markov decision process. In this section, we analyze the properties of the objective function and the optimal update policy based on the Markov decision process. Then we propose a computation algorithm to derive the optimal update policy δ^* using a backward induction method.

A. Analysis of the Optimal Update Policy

Based on Eq. (10) and the theory of Markov decision process, we let

$$V_1 = E(c_1(s_1, d_1) + c_2(s_2, d_2) + \dots + c_M(s_M, d_M))$$

represent the expected total system cost (value function) from time t_1 to time t_M . Similarly,

$$V_2 = E(c_2(s_2, d_2) + \dots + c_M(s_M, d_M)).$$

represents the expected total system cost from time t_2 to time t_M . Thus we have

$$V_1 = c_1(s_1, d_1) + V_2.$$

and more generally,

$$V_m = c_m(s_m, d_m) + V_{m+1}.$$

That is to say, the value function V_m at system time t_m can be considered the sum of the system cost at current decision point t_m and the value function at the next decision point t_{m+1} .

When $m = M$, let $V_{M+1} = 0$ and we get

$$V_M = c_M(s_M, d_M).$$

Because the accumulated purchase data arriving to an ERP system is stochastic, all the possible values of accumulated purchase data quantity at each decision point should be considered. Therefore, we include the probability distribution of system state s_m transiting to

s_{m+1} in the value function. Then, $V_m = c_m(s_m, d_m) + V_{m+1}$ can be rewritten as:

$$V_m = c_m(s_m, d_m) + \sum_{s_{m+1} \in S} P_{s_m, d_m, s_{m+1}} \cdot V_{m+1}. \quad (12)$$

where V_m represents the expected total system cost from system time t_m to t_M .

Based on the principle of Markov decision process, to obtain the minimal expected total system cost V_m^* , we must find a set of optimal update decision δ_m^* under the current system state s_m that can minimize V_m , where $\delta_m^* = (d_m^*, d_{m+1}^*, \dots, d_M^*)$ and $m = 1, 2, \dots, M$. The optimal expected total system cost V_m^* from system time t_m to t_M is defined as:

$$V_m^* = \min(c_m(s_m, d_m) + \sum_{s_{m+1} \in S} P_{s_m, d_m, s_{m+1}} \cdot V_{m+1}^*). \quad (13)$$

When $m = M$, we can get $V_M^* = c_M(s_M, d_M)$ because $V_{M+1} = 0$.

From the above analysis, we know that V_1^* is the minimal expected total system cost from system time t_1 to t_M . Considering that there are infinite number of possible system states at the start decision point t_1 , the optimal expected total system cost for all M inventory inquiries in a time horizon can be represented by:

$$C = \sum_{s_1 \in S} V_1^* \cdot P_{s_1}. \quad (14)$$

where P_{s_1} is the probability that the system state is s_1 at time t_1 .

As mentioned before, the system state s_1 is represented by the quantity of accumulated unprocessed purchase data by time t_1 . Recall that t_1 is the time that the first inventory inquiry arrives at the ERP system. Therefore, the data quantity accumulated by time t_1 equals the data quantity accumulated from the start time of the time horizon to time t_1 . According to Eq. (3), P_{s_1} can be represented by:

$$P_{s_1} = P(I_{0,1} = q) = \frac{(\lambda_u)^q \cdot \lambda_r}{(\lambda_u + \lambda_r)^{q+1}}. \quad (15)$$

In order to obtain the optimal update policy from system time t_m to t_M , we next analyze

the characteristics of V_m^* and V_m . Based on the optimal expected total system cost V_m^* shown in Eq. (13), we define:

$$R(s_m, d_m) = c_m(s_m, d_m) + \sum_{s_{m+1} \in S} P_{s_m, d_m, s_{m+1}} \cdot V_{m+1}^*. \quad (16)$$

Then Eq. (13) can also be rewritten as

$$V_m^* = \min R(s_m, d_m).$$

One of the very important properties of Markov decision process is the optimality of monotone policies. It is obtained for the problem of maximizing an objective function. However, the objective function in this study is to minimize the expected total system cost. Therefore, we redefine our problem and transform the minimum objective function (Eq. (13)) into a maximum one:

$$G_m^* = \max(-c_m(s_m, d_m) + \sum_{s_{m+1} \in S} P_{s_m, d_m, s_{m+1}} \cdot G_{m+1}^*). \quad (17)$$

where G_m^* is the maximal reward from time t_m to t_M and $G_m^* = -V_m^*$.

Based on the optimality of monotone policies in Markov decision process [32], we obtain the following properties for problem (17):

THEOREM 2. For problem (17), let $m = 1, 2, \dots, M - 1$, if:

1. $-c_m(s_m, d_m)$ is nonincreasing in s_m for all $d_m \in D$,
2. $\sum_{s_{m+1} \in S} P_{s_m, d_m, s_{m+1}}$ is nondecreasing in s_m for all $s_{m+1} \in S$ and $d_m \in D$,
3. $-c_m(s_m, d_m)$ is a superadditive function on $S \times D$,
4. $\sum_{s_{m+1} \in S} P_{s_m, d_m, s_{m+1}} \cdot G_{m+1}^*$ is a superadditive function on $S \times D$ for all $s_{m+1} \in S$,

and

5. $-c_M(s_M, d_M)$ is nonincreasing in s_m ,

then the optimal decision policy d_m^* is monotone nondecreasing in system state s_m for $m =$

1,2, ..., M.

We next provide the definition on superadditive (see Definition 1) and prove that each of the condition is required to get the monotone nondecreasing optimal policy. Please refer to the **APPENDIX** for the detailed proof of Theorem 2.

DEFINITION 1. A real-valued function $g(x, y)$ on $X \times Y$ is superadditive if for $x^+ \geq x^-$ in X and $y^+ \geq y^-$ in Y :

$$g(x^+, y^+) + g(x^-, y^-) \geq g(x^+, y^-) + g(x^-, y^+). \quad (18)$$

From Eq. (13) and Eq. (16), we can see that the optimal expected total system cost V_m^* can be interpreted as the comparison result between the expected total system cost under an update decision $R(s_m, d_m = 1)$ and the expected total system cost under a no update decision $R(s_m, d_m = 0)$. If $R(s_m, d_m = 0)$ at system time t_m is equal to or greater than $R(s_m, d_m = 1)$, then updating the ERP system with the newly arriving purchase data is a better choice. Otherwise, it is better not to update the ERP system.

From Theorem 2, we conclude that the optimal update policy d_m^* is monotone nondecreasing in system state s_m . We already know that d_m is composed by 0 and 1. Therefore monotone nondecreasing of d_m^* means that d_m^* is increasing from 0 to 1. That is to say, there is a control limit l_m^* for system state s_m at each decision point determining the optimal update policy is 0 or 1. The optimal update policy can be transformed into the form:

$$d_m^* = \begin{cases} 0 & s_m < l_m^* \\ 1 & s_m \geq l_m^* \end{cases} \quad (19)$$

Based on Eq. (19), if the accumulated purchase data s_m is less than the control limit l_m^* , it is optimal not to update the system, and when the accumulated purchase data is equal to or greater than l_m^* , it is optimal to update the system. Therefore, l_m^* is the minimal quantity of

accumulated purchase data satisfying $R(s_m, d_m = 0) \geq R(s_m, d_m = 1)$ and $l_m^* \in Z^+$. In this way, the problem of finding the optimal update policy δ^* is reduced to that of determining the control limits l^* , where $l^* = (l_1^*, l_2^*, \dots, l_m^*, \dots, l_M^*)$.

B. The Computational Method

Backward induction is an efficient method for solving Markov decision problem with a finite time horizon. Therefore, we design an algorithm to find the optimal update policy l^* for the purchase data update problem based on the backward induction method. As explained before, the objective is to determine the control limit l_m^* at each decision point, which is the minimal quantity of accumulated purchase data satisfying $R(s_m, d_m = 0) \geq R(s_m, d_m = 1)$. After further analyzing $R(s_m, d_m)$ in Eq. (16), we find that it is impossible to directly calculate $R(s_m, d_m = 0) \geq R(s_m, d_m = 1)$ for $m = 1, 2, \dots, M$, because at each step we need to compute V_{m+1}^* for an infinite number of possible system state s_{m+1} . However, from Theorem 2, we know that the optimal decision policy d_m^* is monotone nondecreasing in system state s_m for $m = 1, 2, \dots, M$. Therefore, we can compare and record the values of $R(s_m, d_m = 0)$ and $R(s_m, d_m = 1)$ at each step when s_m gradually increases from zero until the minimal s_m makes $R(s_m, d_m = 0)$ equal or larger than $R(s_m, d_m = 1)$. In fact, the minimal s_m by solving $R(s_m, d_m = 0) \geq R(s_m, d_m = 1)$ is the optimal control limit l_m^* at each decision point. Furthermore, when $m = M$, we have $R(s_M, d_M) = c_M(s_M, d_M)$, and when $m = M - 1, \dots, 2, 1$, we have $R(s_m, d_m) = c_m(s_m, d_m) + \sum_{s_{m+1} \in S} P_{s_m, d_m, s_{m+1}} \cdot V_{m+1}^*$, therefore we should consider the two different update decisions for $R(s_m, d_m)$ separately.

In backward induction, we first obtain l_M^* , which is the minimal quantity of accumulated purchase data at time t_M that satisfies:

$$R(s_M, d_M = 0) \geq R(s_M, d_M = 1). \quad (20)$$

Based on Eqs. (8) and (16), and $V_{M+1}^* = 0$, Eq. (20) is reduced to:

$$c_s(s_M) \geq c_u. \quad (21)$$

Similarly, for $m = M - 1, M - 2, \dots, 2, 1$, we need to check whether $R(s_m, d_m = 0) \geq R(s_m, d_m = 1)$ is satisfied as s_m gradually increases from zero, a procedure that helps us obtain l_m^* at each decision point. The complete algorithm is summarized in Fig. A1 in the **APPENDIX**.

V. EXPERIMENTS AND RESULTS FOR PERFORMANCE EVALUATION

To test the performance and effectiveness of the aperiodic purchase data update policy obtained from the proposed method, we conducted experiments based on a real-life dataset from the Hongmen Advanced Technology Co., Ltd. (or Hongmen for short) located in the Shen Zhen city in China (<http://www.hongmen.com/>). Hongmen has a history of almost 20 years, a registered capital of 66.8 million RMB, more than 550 employees. It is one of the largest gate manufacturers in China, and specializes in developing, designing, and producing over 21 series of products such as automatic gates, IC card intelligent systems, and intelligent parking management systems, etc. By the end of the year 2015, its total sales, cost of goods sold, and net profit have reached RMB257.353 million, RMB165.200 million, and RMB10.846 million, respectively. In May 2011, Hongmen invested more than 2,104,626 RMB to implement the SAP-R/3 ERP system which made up 90.98% of its total IT cost. The variety of products in this company require a wide variety of materials be purchased and managed, which brings a big challenge to maintain the timeliness of inventory data in its ERP system. We use parameter values estimated from real-life data from this company in our experiments.

A. Parameter Settings

In this paper, the unit of time is set to be one day and the entire time horizon is one year or 365 days in our experiments. According to the real operation situation in Hongmen, the arrival rate of new purchase data λ_u , representing the average number of purchase records arriving to the ERP system in one day, was set to 182. The inventory state was checked every seven days in the company, therefore the arrival rate of inventory inquiries λ_r was $1/7$. Based on information provided by the ERP system manager of Hongmen, the update cost c_u consists of computation cost and personnel cost. Computation cost mainly includes the hardware and software cost incurred when updating the ERP system. According to the ERP system operation manager, the average maintenance cost on ERP hardware and software is RMB474,500 per year, and the average monthly salary for the material manager is RMB6,900. Hongmen estimated that the cost of running one update approximately equals one day's of hardware and software maintenance cost and one day's salary of the material manager. Hence, the unit computation cost and personal cost incurred when updating the ERP system were set to RMB1,300 (RMB474,500/365 days) and RMB230 (RMB6,900/30 days), respectively. Therefore, the total update cost c_u was RMB1,530 (RMB1,300+ RMB230).

Regarding the staleness cost, according to Eq. (11) and the earlier discussion, we know that the staleness cost, which takes the form of $c_s(s_m) = (c_p + c_l) \cdot F(s_m)$, is a monotone increasing function, where c_p and c_l are constants. Therefore, the staleness severity function $F(s_m)$ is also a monotone increasing function of its parameter, the system state s_m . Depending on the real-world problem context, $F(s_m)$ can take different functional forms. For ease of analysis, one possibility is to use the cumulative distribution function (CDF) of a certain

distribution to represent the monotonically increasing nature of this function.

In the Hongmen Company, the average purchase cost (setup cost) is RMB1,000, mainly including the travel expense, mailing expense, and so on. The ordering policy in the company is similar to the (R, Q) policy, i.e., when the inventory position declines to or below the reorder point R , a batch quantity of size Q is ordered. Based on the calculation from the ERP database in Hongmen, the average purchase quantity Q is 150 units, the average reorder point R is 50 units, the average holding cost per unit is 20 RMB. Therefore, the average inventory cost is 2,500 RMB $((150/2+50)*20)$. In summary, the staleness cost can be rewritten as $c_s(s_m) = 3500 \cdot F(s_m)$.

B. Performance Analysis of the Update Policy

The performance of the optimal purchase data update policy proposed in this paper is evaluated using the cost savings compared with the traditional data update policies based on the fixed interval. The three fixed interval update policies are described below [28]:

- 1) **Fixed time interval policy.** This can also be called periodical policy which requires that the information system is updated after a fixed time interval, e.g. every five hours.
- 2) **Fixed inquiry interval policy.** Under this policy, an information system should be immediately updated when it receives a fixed number of inquiries, e.g. after every five inquiries received by this system.
- 3) **Fixed update interval policy.** This policy demands that an information system is updated after a fixed number of data updates have arrived at the system, e.g. when the number of accumulated new data is equal to or larger than five records.

We define C , C_{time} , $C_{inquiry}$, and C_{update} as the optimal expected total system costs

obtained by implementing the aperiodic data update policy proposed in this study, the fixed time interval policy, the fixed inquiry interval policy, and the fixed update interval policy, respectively. Similarly, we let t' , i' and u' denote the percentages of cost savings achieved by the proposed aperiodic update policy over the three traditional fixed interval update policies.

In the first set of experimental runs, we let $F(s_m)$ take the functional-form of the CDF of exponential distribution (represented by $F_{exponential}$), i.e.,

$$F(s_m) = F_{exponential}(s_m) = 1 - e^{-\lambda \cdot s_m}. \quad (22)$$

where λ is a parameter of the function. It can be shown that this is a monotone increasing concave function of s_m . To see the performance and advantages of the compared update policies, we conducted a series of experiments while changing the value of the parameter λ . The cost results and the percentage of cost savings achieved by proposed data update policy (in parenthesis) are summarized in TABLE I. For ease of comparison, we also plot the logarithms of expected total costs in Fig. 4.

TABLE I
COMPARISONS OF EXPECTED TOTAL COST UNDER DIFFERENT DATA UPDATE POLICES

	C	$C_{time} (t')$	$C_{inquiry} (i')$	$C_{update} (u')$
$\lambda=0.0001$	40246	53118 (24.2%)	61960 (35%)	53111 (24.2%)
$\lambda=0.0005$	61073	118770 (48.6%)	162980 (62.5%)	118740 (48.6%)
$\lambda=0.001$	67966	167970 (59.5%)	256600 (73.5%)	167900 (59.5%)
$\lambda=0.005$	76544	375590 (79.6%)	963900 (92.1%)	375240 (79.6%)
$\lambda=0.01$	77975	530560 (85.3%)	1844000 (95.8%)	529870 (85.3%)
$\lambda=0.05$	79207	1197200 (93.4%)	9021700 (99.1%)	1191000 (93.3%)
$\lambda=0.1$	79366	1807300 (95.6%)	17564000 (99.5%)	1663400 (95.2%)
$\lambda=0.5$	79491	5807400 (98.6%)	73565000 (99.8%)	3424700 (97.7%)

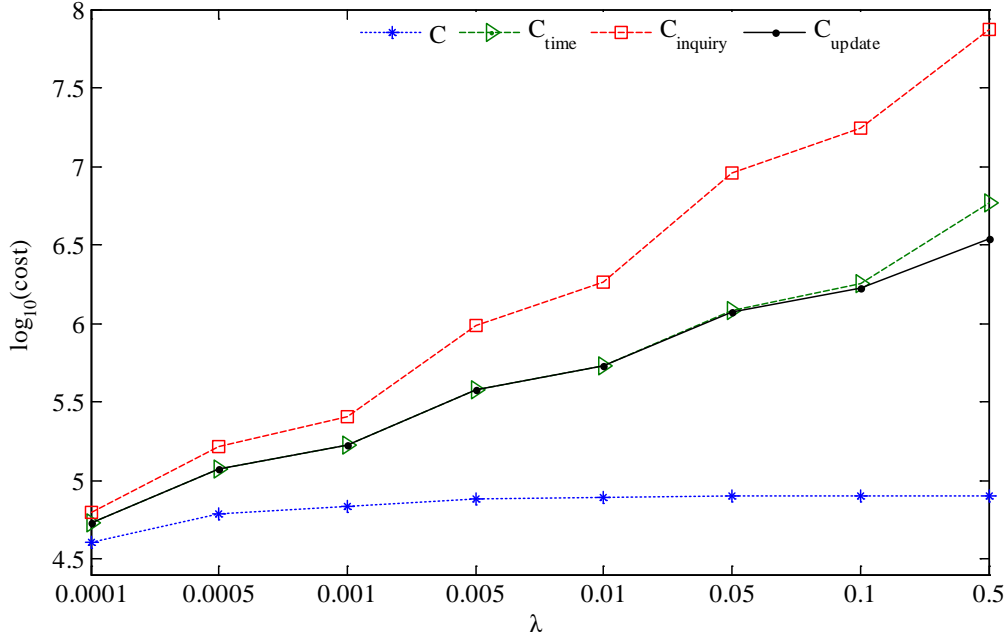


Fig. 4. The logarithm of expected total cost in TABLE I

The results in TABLE I and Fig. 4 show that the proposed aperiodic update policy consistently outperforms all three fixed interval update policies and can lead to significant cost savings. Another apparent result shown in TABLE I and Fig. 4 is that the optimal expected total system cost increases with the value of parameter λ . This is because the higher value of λ leads to a higher value of $F_{exponential}(s_m)$ when system state s_m is fixed. According to Theorem 1, the optimal expected total system cost also increases as a result. For the three fixed interval update policies, when the value of parameter λ gets higher, the system becomes more sensitive to the stale data and it has to be updated at a smaller interval. Therefore, the optimal expected total system cost is also increasing with λ . Another noticeable result demonstrated in TABLE I and Fig. 4 is that among those three fixed interval update policies, the fixed update interval policy is better than the fixed time interval policy, which is in turn better than the fixed inquiry interval policy, i.e. $C_{update} < C_{time} < C_{inquiry}$. This is consistent with the findings of prior research [28].

Additional experiments were conducted to examine the effect of different forms of staleness severity functions on the performances of different update policies. Specifically, in addition to the *CDF* of exponential distribution, we tried two other monotone increasing functions with different curves. The logistic function (denoted by $F_{logistic}$) shown in Eq. (23) represent a monotone increasing *S* curve:

$$F(s_m) = F_{logistic}(s_m) = \frac{1}{1+e^{(\alpha-\beta \cdot s_m)}}. \quad (23)$$

The *CDF* of uniform distribution (denoted by $F_{uniform}$) is a monotone increasing piecewise linear curve:

$$F(s_m) = F_{uniform}(s_m) = \begin{cases} 0, & s_m < 0, \\ \frac{s_m}{\delta}, & 0 \leq s_m < \delta, \\ 1, & s_m \geq \delta. \end{cases} \quad (24)$$

As explained before, we need $F(s_m) = 0$ when $s_m = 0$. Although theoretically the value of α in Eq. (23) needs to be infinite to meet this requirement, we set the value of α to 15 because such a value can already make $F(s_m)$ incredibly close to zero (0.0000003059). The comparison of the optimal expected total system costs under four data update policies and changing parameter values in two different staleness severity functions are shown in TABLE II.

TABLE II
COMPARISONS OF EXPECTED TOTAL COST UNDER DIFFERENT UPDATE POLICIES AND STALENESS SEVERITY FUNCTIONS

	Parameters	C	$C_{time}(t')$	$C_{inquiry}(i')$	$C_{update}(u')$
	$\beta = 0.001$	7429.6	33154 (77.6%)	36598 (79.7%)	33151 (77.6%)
	$\beta = 0.005$	26907	74205 (63.7%)	91460 (70.6%)	74191 (63.7%)
	$\beta = 0.01$	39758	104940 (62.1%)	139450 (71.5%)	104910 (62.1%)
	$\beta = 0.05$	65862	234650 (71.9%)	424850 (84.5%)	234510 (71.9%)

$\beta = 0.1$	72004	335920 (78.6%)	786980 (90.9%)	335640 (78.5%)
$\beta = 0.5$	77889	786320 (90.1%)	3954900 (98%)	784800 (90.1%)
$\beta = 1$	78700	1112000 (92.9%)	7829900 (98.9%)	1109000 (92.9%)
$\delta = 100$	79714	609080 (86.9%)	2404800 (96.7%)	608170 (86.9%)
$\delta = 500$	75084	272390 (72.4%)	544790 (86.2%)	272210 (72.4%)
$\delta = 1000$	70502	192610 (63.4%)	312280 (77.4%)	192520 (63.4%)
$\delta = 3000$	58833	111200 (47.1%)	149950 (60.8%)	111170 (47.1%)
$\delta = 5000$	51963	86137 (39.7%)	109390 (52.5%)	86119 (39.7%)
$\delta = 8000$	45328	68097 (33.4%)	82629 (45.1%)	68086 (33.4%)
$\delta = 10000$	42183	60908 (30.7%)	72533 (41.8%)	60899 (30.7%)
$\delta = 15000$	36625	49728 (26.3%)	57477 (36.3%)	49722 (26.3%)

Similar to Fig. 4, we plot the logarithms of expected total costs in Fig. 5 and Fig. 6, corresponding to logistic and uniform functions, respectively.

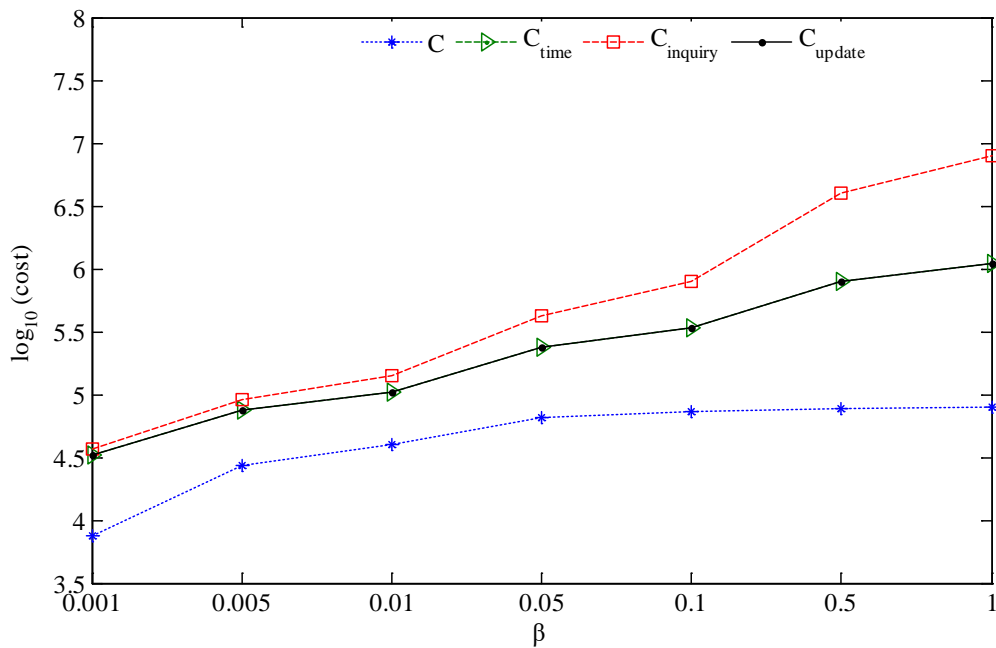


Fig. 5. The logarithm of expected total cost with $F_{logistic}$

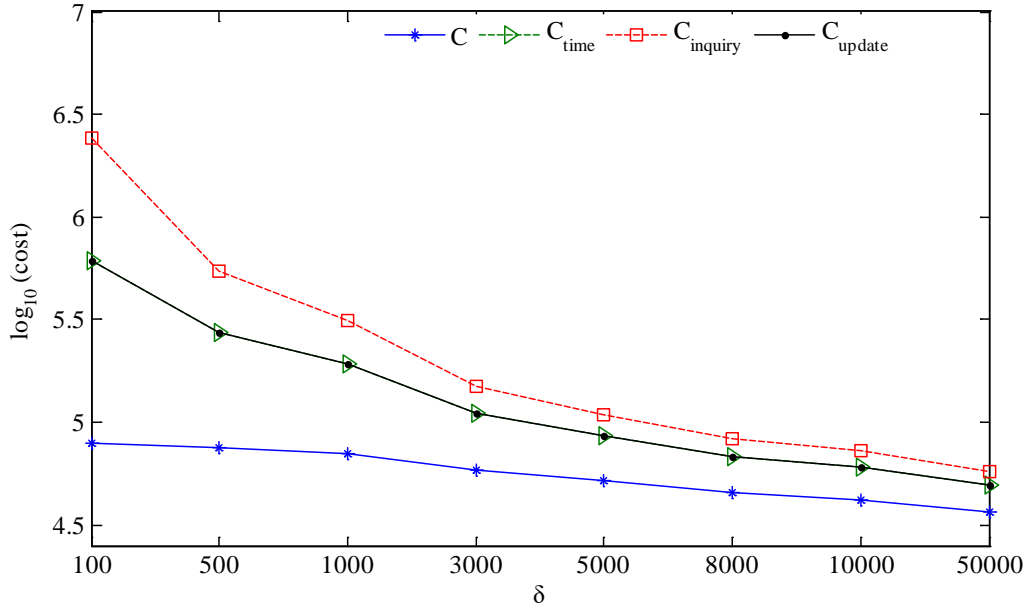


Fig. 6. The logarithm of expected total cost with $F_{uniform}$

Once again, the experimental results summarized in TABLE II and Figs. 5-6 show that the update policy proposed in this research consistently and significantly outperforms other three fixed interval update policies. When $F(s_m)$ takes the logistic functional form, the optimal expected total costs of all the four data update policies increase with the value of β . This is because the value of $F_{logistic}$ is higher with a larger β value. When $F(s_m)$ takes the form of the *CDF* of uniform distribution, the expected total costs of all four data update policies decrease as the δ value increases. This is because with other factors fixed, the value of $F_{uniform}(s_m)$ decreases as the value of δ increases. From TABLE II and Figs. 5-6, we also observe that among the three fixed interval update policies, the fixed inquiry interval policy always incurs a higher cost than the fixed time interval policy and the fixed update interval policy, and the cost incurred by the fixed time interval policy is slightly higher than the fixed update interval policy. This result is also consistent with the previous study [28].

VI. CONCLUSIONS AND DISCUSSIONS

Timeliness is one of the most important quality dimensions of data assets managed by organizations. However, in the era of big data, with the sheer volume of constantly changing data, how to achieve and maintain data timeliness in dynamic database systems has become a challenging yet unresolved operational issue to many organizations. This research tries to address this issue by helping organizations decide when and how frequently to update data in their database systems. We model the data update problem as a Markov decision process and design a dynamic planning algorithm to obtain the optimal update policy based on the backward induction method. A set of experiments, using three different types of staleness severity functions, are conducted to verify the performance of the proposed aperiodic data update policy. The results show that the proposed data update policy consistently and significantly outperforms the traditional fixed interval data update policies.

This study has important practical implications for organizations. First, although fixed interval update policies might be easier to implement, given the amount of cost savings that can be achieved by the proposed aperiodic database update policy, organizations should explore the possibility of using aperiodic update policies to save costs. Second, the staleness severity function and other parameter values have a substantial impact on the optimal update policy. For this reason, efforts should be taken to make sure that the staleness costs are accurately estimated. Third, although we illustrate the proposed update policy using ERP systems, we believe that the aperiodic update policy, with some adaptation, could be used for the maintenance of a wide range of data assets and applications such as patient medical record maintenance, GIS data update, and website maintenance, etc.

There are a few possible directions for future research, all involving improving or extending the aperiodic policy proposed in the present research. First, the proposed policy is developed for updating reliable and structured data such as purchase data. In other more complex general setting, data could arrive from multiple sources with varying degrees of reliability. One could extend the data update policy proposed in this study for processing and updating data from such more complex data sources. Second, the present study assumes that there is only one type of inquiries, the derived policy could be extended to cover scenarios where there is more than one type of inquiries. Third, it should be possible to derive a data update policy for more general scenarios where the arrival rate of purchase data or inventory inquiries changes with time, i.e., the parameters λ_u and λ_r are functions of time.

Acknowledgement

This work was funded by the National Natural Science Foundation under Grant No. 71428003, 71471144, and 71071126. The authors also acknowledge the support from The Key Lab of the Ministry of Education for Process Control & Efficiency Engineering and Experiment Center for the Teaching of Management Education in Xi'an Jiaotong University. The third author is the corresponding author.

REFERENCES

- [1] O. Zach, B. E. Munkvold, and D. H. Olsen, "ERP system implementation in SMEs: exploring the influences of the SME context," *Enterprise Information Systems*, vol. 8, no. 2, pp. 309-335, 2014.
- [2] M. G. Morris, and V. Venkatesh, "Job characteristics and job satisfaction: understanding

- the role of enterprise resource planning system implementation,” *MIS Quarterly*, vol. 34, no. 1, pp. 143-161, 2010.
- [3] Y. Zeng, and M. J. Skibniewski, “Risk assessment for enterprise resource planning (ERP) system implementations: a fault tree analysis approach,” *Enterprise Information Systems*, vol. 7, no. 3, pp. 332-353, 2013.
- [4] Y.-Y. Huang, and R. B. Handfield, “Measuring the benefits of ERP on supply management maturity model: a “big data” method,” *International Journal of Operations & Production Management*, vol. 35, no. 1, pp. 2-25, 2015.
- [5] R. Basu, P. Upadhyay, M. C. Das *et al.*, “An approach to identify issues affecting ERP implementation in Indian SMEs,” *Journal of Industrial Engineering and Management*, vol. 5, no. 1, pp. 133-154, 2012.
- [6] W.-H. Tsai, Y.-W. Chou, J.-D. Leu *et al.*, “Investigation of the mediating effects of IT governance-value delivery on service quality and ERP performance,” *Enterprise Information Systems*, vol. 9, no. 2, pp. 139-160, 2015.
- [7] J. Manyika, M. Chui, B. Brown *et al.*, *Big Data: The Next Frontier for Innovation, Competition, and Productivity*: McKinsey Global Institute, 2011.
- [8] W. Zong, F. Wu, L.-K. Chu *et al.*, “A discriminative and semantic feature selection method for text categorization,” *International Journal of Production Economics*, vol. 165, pp. 215-222, 2015.
- [9] J. Lu, H. Pham, H. Zhu *et al.*, “A Novel Indexing Method for Improving Timeliness of High-Dimensional Data,” in *Twentieth Americas Conference on Information Systems*, Savannah, USA, 2014, pp. 1-12.

- [10] M. Bouzeghoub, and V. Peralta, "A Framework for Analysis of Data Freshness," in the 2004 International Workshop on Information Quality in Information Systems, Paris, France, 2004, pp. 59-67.
- [11] X. Fang, and R. Rachamadugu, "Policies for knowledge refreshing in databases," *Omega*, vol. 37, no. 1, pp. 16-28, 2009.
- [12] M. S. Islam, "Regulators of Timeliness Data Quality Dimension for Changing Data Quality in Information Manufacturing System (IMS)," in The Third International Conference on Digital Information Processing and Communications, United Arab Emirates, 2013, pp. 126-133.
- [13] B. Heinrich, and M. Klier, "Metric-based data quality assessment — Developing and evaluating a probability-based currency metric," *Decision Support Systems*, vol. 72, pp. 82-96, 2015.
- [14] Y. Wand, and R. Y. Wang, "Anchoring data quality dimensions in ontological foundations," *Communications of the ACM*, vol. 39, no. 11, pp. 86-95, 1996.
- [15] Y. W. Lee, and D. M. Strong, "Knowing-why about data processes and data quality," *Journal of Management Information Systems*, vol. 20, no. 3, pp. 13-39, Win, 2003.
- [16] D. Ballou, R. Wang, H. Pazer *et al.*, "Modeling information manufacturing systems to determine information product quality," *Management Science*, vol. 44, no. 4, pp. 462-484, 1998.
- [17] I. C. Italiano, and J. E. Ferreira, "Synchronization options for data warehouse designs," *Computer*, vol. 39, no. 3, pp. 53-57, 2006.
- [18] M. Mannino, S. N. Hong, and I. J. Choi, "Efficiency evaluation of data warehouse

- operations,” *Decision Support Systems*, vol. 44, no. 4, pp. 883-898, 2008.
- [19] J. Wang, J. Lu, Z. Fang *et al.*, “PL-Tree: An Efficient Indexing Method for High-Dimensional Data,” in the 13th International Symposium on Advances in Spatial and Temporal Databases, Munich, Germany, 2013, pp. 183-200.
- [20] Q. Han, and N. Venkatasubramanian, “Addressing Timeliness/Accuracy/Cost Tradeoffs in Information Collection for Dynamic Environments,” in the 24th IEEE International Real-Time Systems Symposium, Cancun, Mexico, 2003, pp. 108-117.
- [21] J. R. Davis, “The challenges of real time data management,” *BeyeNETWORK Custom Research Report*, pp. 1-12, 2010.
- [22] C. Chen, and J. Chen, “Timeliness evaluation of task-oriented networked space-based information system,” *Journal of Systems Engineering and Electronics*, vol. 22, no. 4, pp. 621-627, 2011.
- [23] K. M. Chandy, J. C. Browne, C. W. Dissly *et al.*, “Analytic models for rollback and recovery strategies in data base systems,” *IEEE Transactions on Software Engineering*, vol. SE-1, no. 1, pp. 100-110, 1975.
- [24] J. Srivastava, and D. Rotem, “Analytical Modeling of Materialized View Maintenance,” in the Seventh ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Austin, USA, 1988, pp. 126-134.
- [25] A. Segev, and W. P. Fang, “Optimal update policies for distributed materialized views,” *Management Science*, vol. 37, no. 7, pp. 851-870, Jul, 1991.
- [26] B. Adelberg, H. Garcia-Molina, and B. Kao, “Applying Update Streams in a Soft Real-Time Database System,” in the 1995 ACM SIGMOD International Conference on

Management of Data, San Jose, USA, 1995, pp. 245-256.

- [27] Y. B. Ling, and J. Mi, “An optimal trade-off between content freshness and refresh cost,” *Journal of Applied Probability*, vol. 41, no. 3, pp. 721-734, Sep, 2004.
- [28] D. Dey, Z. Zhang, and P. De, “Optimal synchronization policies for data warehouses,” *INFORMS Journal on Computing*, vol. 18, no. 2, pp. 229-242, 2006.
- [29] M. V. Mannino, and Z. Walter, “A framework for data warehouse refresh policies,” *Decision Support Systems*, vol. 42, no. 1, pp. 121-143, 2006.
- [30] X. Fang, O. R. L. Sheng, and P. Goes, “When Is the Right Time to Refresh Knowledge Discovered from Data?,” *Operations Research*, vol. 61, no. 1, pp. 32-44, 2013.
- [31] V. Paxson, and S. Floyd, “Wide area traffic - The failure of poisson modeling,” *IEEE-ACM Transactions on Networking*, vol. 3, no. 3, pp. 226-244, Jun, 1995.
- [32] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, New York: John Wiley & Sons, 2009.

APPENDIX

1. Key Notations.

TABLE AI
SUMMARY OF NOTATIONS

Parameter	Description on the parameter
N	The time horizon of the purchase data update problem.
M	The average total number of inventory inquiries arriving at an ERP system in a time horizon.
r_m	The m th inventory inquiry arriving at an ERP system, $m = 1, 2, \dots, M$, $m \in Z^+$.
t_m	The time when the m^{th} inventory inquiry arrives at an ERP system.
s_m	The system state at time t_m , representing the accumulated quantity of unprocessed purchase data at time t_m and $s_m \in Z^+$.
S	The set of system state, $s_m \in S$.
D	The decision set.
d_m	The decision made when the m^{th} inventory inquiry arrives at an ERP system at time t_m , $d_m \in D = \{0, 1\}$.
$P_{s_m, d_m, s_{m+1}}$	State transition probability, representing the probability that the system state s_m at time t_m under decision d_m transits to the system state s_{m+1} at time t_{m+1} .
λ_u	The arrival rate of new purchase data in an ERP system.
λ_r	The arrival rate of an inventory inquiry in an ERP system.
$I_{m-1, m}$	The purchase data accumulated between time interval t_{m-1} and t_m .
c_u	The update cost at time t_m .
$c_s(s_m)$	The staleness cost for inventory inquiry r at time t_m as a function of s_m .
$F(s_m)$	The data staleness severity function.
$c_m(s_m, d_m)$	The system cost at time t_m under decision d_m .
C	The optimal expected total system cost for M inventory inquiries in a time horizon.
V_m	The expected total system cost from time t_m to t_M , $V_m \geq 0$.
δ	The update policy in a time horizon, i.e. the update decision made at each decision point, $\delta = (d_1, d_2, \dots, d_M)$.
δ_m^*	The optimal update policy from time t_m to t_M , $\delta_m^* = (d_m^*, d_{m+1}^*, \dots, d_M^*)$.
l_m^*	The optimal control limit at time t_m , $l_m^* \in Z^+$.
l^*	The optimal control limit for system state, where $l^* = (l_1^*, l_2^*, \dots, l_m^*, \dots, l_M^*)$.

2. Proof of Theorem 1:

PROOF. Let $F_1(s_m) \geq F_2(s_m)$. According to Eq. (9) and Eq. (10), the optimal expected

total system cost can be written as $C = \min E(c_1(s_1, d_1) + c_2(s_2, d_2) + \dots + c_M(s_M, d_M))$.

Let C' and C'' represent the expected total system cost by applying data staleness function $F_1(s_m)$ and $F_2(s_m)$, respectively. $d_m^{*'}$ and $d_m^{*''}$ denote the optimal update policy at time t_m under different staleness functions. Therefore we can get:

$$C' = \min E(c_1(s_1, d_1)' + c_2(s_2, d_2)' + \dots + c_M(s_M, d_M)')$$
 and

$$C'' = \min E(c_1(s_1, d_1)'' + c_2(s_2, d_2)'' + \dots + c_M(s_M, d_M)'').$$

For $c_1(s_1, d_1)'$ and $c_1(s_1, d_1)''$, when $d_1^{*'} = d_1^{*''} = 0$, according to Eq. (8) and Eq. (11), we have $c_1(s_1, d_1)' = (c_p + c_l) \cdot F_1(s_m)$ and $c_1(s_1, d_1)'' = (c_p + c_l) \cdot F_2(s_m)$. Therefore we can get $c_1(s_1, d_1)' \geq c_1(s_1, d_1)''$ for $F_1(s_m) \geq F_2(s_m)$.

When $d_1^{*'} = d_1^{*''} = 1$, according to Eq. (8) and Eq. (11), we have $c_1(s_1, d_1)' = c_u$ and $c_1(s_1, d_1)'' = c_u$. Therefore we can get $c_1(s_1, d_1)' = c_1(s_1, d_1)''$ for $F_1(s_m) \geq F_2(s_m)$.

When $d_1^{*'} = 1$ and $d_1^{*''} = 0$, according to Eq. (8) and Eq. (11), we have $c_1(s_1, d_1)' = c_u$ and $c_1(s_1, d_1)'' = (c_p + c_l) \cdot F_2(s_m)$. $d_1^{*''} = 0$ indicates that the data staleness cost is smaller than the update cost at time t_1 , i.e., $c_1(s_1, d_1)'' = (c_p + c_l) \cdot F_2(s_m) < c_u$. Therefore we have $c_1(s_1, d_1)' > c_1(s_1, d_1)''$ for $F_1(s_m) \geq F_2(s_m)$.

At last, when $d_1^{*'} = 0$ and $d_1^{*''} = 1$, according to Eq. (8) and Eq. (11), we can conclude that $c_1(s_1, d_1)' = (c_p + c_l) \cdot F_1(s_m)$ and $c_1(s_1, d_1)'' = c_u$. $d_1^{*'} = 0$ indicates that the data staleness cost is smaller than the data update cost, i.e.,:

$$(c_p + c_l) \cdot F_1(s_m) < c_u \tag{A1}$$

While $d_1^{*''} = 1$ indicates that the data staleness cost is equal to or larger than the data update cost, i.e.,

$$(c_p + c_l) \cdot F_2(s_m) \geq c_u \tag{A2}$$

Based on Eq. (A1) and Eq. (A2), we know that $F_2(s_m) > F_1(s_m)$ which contradicts the assumption $F_1(s_m) \geq F_2(s_m)$. That is to say, the condition when $d_1^{*'} = 0$ and $d_1^{*''} = 1$ is false and therefore no need to be considered for the next analysis.

Therefore we can get $c_1(s_1, d_1)' \geq c_1(s_1, d_1)''$ for $F_1(s_m) \geq F_2(s_m)$.

By the above analogy, we can get that $c_2(s_2, d_2)' \geq c_2(s_2, d_2)''$, $c_3(s_3, d_3)' \geq c_3(s_3, d_3)''$ until $c_M(s_M, d_M)' \geq c_M(s_M, d_M)''$. Therefore $C' \geq C''$ for $F_1(s_m) \geq F_2(s_m)$.

In summary, the optimal expected total system cost is increasing in the value of staleness function $F(s_m)$ when s_m is fixed.

3. Proof of Theorem 2:

1) $-c_m(s_m, d_m)$ is nonincreasing in s_m for all $d_m \in D$.

PROOF. When $d_m = 0$, in Eq. (8), we know $-c_m(s_m, d_m = 0) = -c_s(s_m)$. According to Eq. (11), $c_s(s_m)$ is an increasing function in system state s_m , therefore $-c_s(s_m, d_m)$ is nonincreasing when $d_m = 0$. When $d_m = 1$, $-c_m(s_m, d_m = 1) = -c_u$, where c_u is a constant. In summary, $-c_m(s_m, d_m)$ is nonincreasing in s_m for all $d_m \in D$.

2) $\sum_{s_{m+1} \in S} P_{s_m, d_m, s_{m+1}}$ is nondecreasing in s_m for all $s_{m+1} \in S$ and $d_m \in D$.

PROOF. Let $s_m = a_m$, $s_{m+1} = a_n$. Due to the different cases for d_m , the proof process should be conducted respectively.

When $d_m = 0$, according to Eq. (5), we have:

$$\begin{aligned} \sum_{s_{m+1} \in S} P_{s_m, d_m=0, s_{m+1}} &= \sum_{s_{m+1} \in S} \frac{(\lambda_u)^{(s_{m+1}-s_m)} \cdot \lambda_r}{(\lambda_u + \lambda_r)^{(s_{m+1}-s_m+1)}} \\ &= \sum_{s_{m+1}=a_n}^{\infty} \frac{(\lambda_u)^{(a_n-a_m)} \cdot \lambda_r}{(\lambda_u + \lambda_r)^{(a_n-a_m+1)}} \\ &= \frac{\lambda_r}{(\lambda_u + \lambda_r)} \cdot \sum_{s_{m+1}=a_n}^{\infty} \left(\frac{\lambda_u}{\lambda_u + \lambda_r} \right)^{(a_n-a_m)} \end{aligned}$$

$$\begin{aligned}
&= \frac{\lambda_r}{(\lambda_u + \lambda_r)} \cdot \frac{\left(\frac{\lambda_u}{\lambda_u + \lambda_r}\right)^{(a_n - a_m)}}{1 - \frac{\lambda_u}{\lambda_u + \lambda_r}} \\
&= \left(\frac{\lambda_u}{\lambda_u + \lambda_r}\right)^{(a_n - a_m)}
\end{aligned}$$

(A3)

Eq. (A3) can be rewritten as $\left(\frac{\lambda_u + \lambda_r}{\lambda_u}\right)^{(a_m - a_n)}$, where $\frac{\lambda_u + \lambda_r}{\lambda_u} > 1$ and $s_m = a_m$.

Therefore, Eq. (A3) is nondecreasing in s_m when $d_m = 0$.

When $d_m = 1$, according to Eq. (7), we have: $\sum_{s_{m+1} \in S} P_{s_m, d_m=1, s_{m+1}} =$

$$\begin{aligned}
&\sum_{s_{m+1} \in S} \frac{(\lambda_u)^{s_{m+1} \cdot \lambda_r}}{(\lambda_u + \lambda_r)^{(s_{m+1} + 1)}} \\
&= \sum_{s_{m+1} = a_n}^{\infty} \frac{(\lambda_u)^{a_n \cdot \lambda_r}}{(\lambda_u + \lambda_r)^{(a_n + 1)}} = \frac{\lambda_r}{(\lambda_u + \lambda_r)} \cdot \\
&\sum_{s_{m+1} = a_n}^{\infty} \left(\frac{\lambda_u}{\lambda_u + \lambda_r}\right)^{a_n} \\
&= \frac{\lambda_r}{(\lambda_u + \lambda_r)} \cdot \frac{\left(\frac{\lambda_u}{\lambda_u + \lambda_r}\right)^{a_n}}{1 - \frac{\lambda_u}{\lambda_u + \lambda_r}} \\
&= \left(\frac{\lambda_u}{\lambda_u + \lambda_r}\right)^{a_n} \tag{A4}
\end{aligned}$$

There is no a_m in Eq. (A4), therefore Eq. (A4) is a constant when $d_m = 1$.

Based on the above analysis, $\sum_{s_{m+1} \in S} P_{s_m, d_m, s_{m+1}}$ is nondecreasing in s_m for all $s_{m+1} \in S$ and $d_m \in D$.

3) $-c_m(s_m, d_m)$ is a superadditive function on $S \times D$.

PROOF. Assume $s^+ \geq s^-$. First, when $d^+ > d^-$, i.e. $d^+ = 1$ and $d^- = 0$, according to Definition 2 and Eq. (8), we have:

$$[-c_m(s^+, d^+)] + [-c_m(s^-, d^-)] = -c_u + [-c_s(s^-)]$$

and

$$[-c_m(s^+, d^-)] + [-c_m(s^-, d^+)] = -c_s(s^+) + (-c_u)$$

$c_s(s_m)$ is increasing in s_m , therefore $c_s(s^+) \geq c_s(s^-)$ for $s^+ \geq s^-$. Hence

$[-c_m(s^+, d^+)] + [-c_m(s^-, d^-)] \geq [-c_m(s^+, d^-)] + [-c_m(s^-, d^+)]$ for $s^+ \geq s^-$ and $d^+ > d^-$, which means $-c_m(s_m, d_m)$ is a superadditive function.

When $d^+ = d^- = 0$, according to Definition 2 and Eq. (8), we have:

$$[-c_m(s^+, d^+)] + [-c_m(s^-, d^-)] = [-c_s(s^+)] + [-c_s(s^-)]$$

and

$$[-c_m(s^+, d^-)] + [-c_m(s^-, d^+)] = [-c_s(s^+)] + [-c_s(s^-)]$$

Therefore, $[-c_m(s^+, d^+)] + [-c_m(s^-, d^-)] = [-c_m(s^+, d^-)] + [-c_m(s^-, d^+)]$ for $s^+ \geq s^-$ and $d^+ = d^- = 0$, which means $-c_m(s_m, d_m)$ is a superadditive function.

Last when $d^+ = d^- = 1$, according to Definition 2 and Eq. (8), we have:

$$[-c_m(s^+, d^+)] + [-c_m(s^-, d^-)] = (-c_u) + (-c_u)$$

and

$$[-c_m(s^+, d^-)] + [-c_m(s^-, d^+)] = (-c_u) + (-c_u)$$

Therefore, $[-c_m(s^+, d^+)] + [-c_m(s^-, d^-)] = [-c_m(s^+, d^-)] + [-c_m(s^-, d^+)]$ for $s^+ \geq s^-$ and $d^+ = d^- = 1$, which means $-c_m(s_m, d_m)$ is a superadditive function.

In summary, $-c_m(s_m, d_m)$ is a superadditive function on $S \times D$.

4) $\sum_{s_{m+1} \in S} P_{s_m, d_m, s_{m+1}} \cdot G_{m+1}^*$ is a superadditive function on $S \times D$ for all $s_{m+1} \in S$.

PROOF. Assume $s^+ \geq s^-$ and $d^+ \geq d^-$. According to Definition 2, to prove that

$\sum_{s_{m+1} \in S} P_{s_m, d_m, s_{m+1}} \cdot G_{m+1}^*$ is a superadditive function, we must prove that:

$$\begin{aligned} & \sum_{s_{m+1} \in S} P_{s^+, d^+, s_{m+1}} \cdot G_{m+1}^* + \sum_{s_{m+1} \in S} P_{s^-, d^-, s_{m+1}} \cdot G_{m+1}^* \\ & \geq \sum_{s_{m+1} \in S} P_{s^+, d^-, s_{m+1}} \cdot G_{m+1}^* + \sum_{s_{m+1} \in S} P_{s^-, d^+, s_{m+1}} \cdot G_{m+1}^* \end{aligned}$$

G_{m+1}^*

When $d^+ > d^-$, i.e. $d^+ = 1$ and $d^- = 0$, by Definition 2 and Eq. (A3) and Eq. (A4),

we have:

$$\sum_{s_{m+1} \in S} P_{s^+, d^+=1, s_{m+1}} + \sum_{s_{m+1} \in S} P_{s^-, d^-=0, s_{m+1}} = \left(\frac{\lambda_u}{\lambda_u + \lambda_r} \right)^{s_{m+1}} + \left(\frac{\lambda_u}{\lambda_u + \lambda_r} \right)^{(s_{m+1} - s^-)}$$

and

$$\sum_{s_{m+1} \in S} P_{s^+, d^-=0, s_{m+1}} + \sum_{s_{m+1} \in S} P_{s^-, d^+=1, s_{m+1}} = \left(\frac{\lambda_u}{\lambda_u + \lambda_r} \right)^{(s_{m+1} - s^+)} + \left(\frac{\lambda_u}{\lambda_u + \lambda_r} \right)^{s_{m+1}}$$

We have proved that $\sum_{s_{m+1} \in S} P_{s_m, d_m, s_{m+1}}$ is nondecreasing in s_m , therefore for $s^+ \geq s^-$,

we can get:

$$\begin{aligned} & \sum_{s_{m+1} \in S} P_{s^+, d^+=1, s_{m+1}} + \sum_{s_{m+1} \in S} P_{s^-, d^-=0, s_{m+1}} \\ & \leq \sum_{s_{m+1} \in S} P_{s^+, d^-=0, s_{m+1}} + \sum_{s_{m+1} \in S} P_{s^-, d^+=1, s_{m+1}} \end{aligned} \quad (A5)$$

Eq. (A5) can be rewritten as:

$$\begin{aligned} & - \sum_{s_{m+1} \in S} (P_{s^+, d^+=1, s_{m+1}} + P_{s^-, d^-=0, s_{m+1}}) \\ & \geq - \sum_{s_{m+1} \in S} (P_{s^+, d^-=0, s_{m+1}} + P_{s^-, d^+=1, s_{m+1}}) \end{aligned}$$

We know $V_m^* \geq 0$ for all $s_m \in S$, therefore we can get:

$$\begin{aligned} & - \sum_{s_{m+1} \in S} (P_{s^+, d^+=1, s_{m+1}} + P_{s^-, d^-=0, s_{m+1}}) V_{m+1}^* \\ & \geq - \sum_{s_{m+1} \in S} (P_{s^+, d^-=0, s_{m+1}} + P_{s^-, d^+=1, s_{m+1}}) V_{m+1}^* \end{aligned} \quad (A6)$$

Since $G_m^* = -V_m^*$, therefore Eq. (A6) can be rewritten as:

$$\begin{aligned} & \sum_{s_{m+1} \in S} P_{s^+, d^+, s_{m+1}} \cdot G_{m+1}^* + \sum_{s_{m+1} \in S} P_{s^-, d^-, s_{m+1}} \cdot G_{m+1}^* \\ & \geq \sum_{s_{m+1} \in S} P_{s^+, d^-, s_{m+1}} \cdot G_{m+1}^* + \sum_{s_{m+1} \in S} P_{s^-, d^+, s_{m+1}} \cdot \end{aligned}$$

G_{m+1}^*

Therefore, $\sum_{s_{m+1} \in S} P_{s_m, d_m, s_{m+1}} \cdot G_{m+1}^*$ is a superadditive function on $S \times D$ for all $s_{m+1} \in S$.

5) $-c_M(s_M, d_M)$ is nonincreasing in s_m .

PROOF. According to Eq. (8), when $m = M$ and $d_M = 0$, $-c_M(s_M, d_M = 0) = -c_s(s_M)$.

According to the analysis before, $c_s(s_m)$ is an increasing function in system state s_m , therefore $-c_s(s_M, d_M)$ is nonincreasing when $d_M = 0$. When $d_M = 1$, $-c_M(s_M, d_M = 1) = -c_u$, where c_u is a constant. In summary, $-c_M(s_M, d_M)$ is nonincreasing in s_m for all $d_m \in D$.

4. Algorithms for obtaining the control limits.

```

Function getlm()
m = M;
l_M^* = ceil(c_s(s_M) ≥ c_u);
for m = M - 1 to 1
  if m = M - 1
    RSM0 = 0;
    v = 0;
    rsm = 0;
    tmp = 1;
    while tmp > 10e - 10 || rsm == 0
      if v < l_M^*
        rsm = c_s(v);
      else
        rsm = c_u;
      end
      tmp = P_{s_m=v, d_m=0, s_{m+1}} · rsm;
      RSM0 = RSM0 + tmp;
      v = v + 1;
    end
  end
  RSM1 = RSM0 + c_u;
  s_m = 0;
  s_n = s_m + 1;
  while s_n > 0
    RSM = 0;
    tmp = RSM0 - c_s(s_m);
    tmp = tmp - (λ_r/λ_u + λ_r) * V_{m+1}^*(s_{m+1} = j);
    tmp = tmp * (λ_u + λ_r) / λ_u;
    tmp = tmp + c_s(s_m + 1);
    RSM = tmp;
    if RSM ≥ RSM1
      R = RSM1;
      JSM = [JSM, R];
      l_m^* = s_m;
      break;
    else
      R = RSM;
      JSM = [JSM, R];
      R1 = RSM;
      s_m = s_m + 1;
      s_n = s_n + 1;
    end
  end
end
end
end

```

Fig. A1. Algorithms for obtaining the control limits.