Abstract

Time studies have been proven useful towards understanding and improving business processes. Particularly, optimizing temporal performance for business processes can directly help reduce costs and bring profits for enterprises. In this paper, we study the properties of temporal performance and develop approaches to optimize it in workflow-based business processes. By applying our data-centric business process modeling techniques, we explore the possibility of reconstructing business process that is realized by modifying the execution order of business activities. Accordingly, we develop efficient algorithms to automatically detect these possibilities and approach optimal temporal performance for business processes. To evaluate our algorithms, a symbolic business process generator, namely G-DCBP, is introduced to stochastically generate symbolic data-centric business processes that can be used to analyze their properties and evaluate the time-optimization approaches according to end-users' specification. The evaluation results validate the efficacy of the algorithms under different scenarios.

Keywords: temporal performance, data-centric, business process optimization, process evaluation

1. INTRODUCTION

In the realm of business process modeling, optimization refers to the automated improvement of business processes using pre-specified quantitative measures of performance (objectives) (Virgidis et al., 2008). Some researchers claim that the ultimate goals of research for business process modeling should optimize business processes and eventually generate improved ones (Virgidis et al., 2008; Aalst et al., 2003). However, in the past decades, while most of research approaches about business process modeling have been proposed to cover modeling techniques and qualitative analysis, business process optimization, e.g. optimizing temporal performance, has gained less attention.

One possible reason is that time sometimes is perceived only as a carrier of resources and the relevant studies have been mostly focused on reducing time to improve the specified product qualities (Zhou et al., 2003; Reijers, 2002). It is nevertheless argued that this multi-aspect optimization might be complicated and result in conflicting criteria (Hofacker et al., 2001). Some of these researches are based on scheduling, that mathematically models the optimal resource allocation of business processes in terms of specified objectives (Floudas et al., 2005; Rommelfanger, 2004; Pinto et al., 1998; Mendez et al., 2006). While these approaches are considered to optimize time-relevant business applications, it is argued by Ernst et al. (2004) that they can only be applied in simplified and ideal business processes where their constraints and objectives can be mathematically modeled. However, some time-relevant business elements, such as decisions, are hardly mathematically expressed. Moreover, modeling techniques for these elements might not be applied when their constraints vary in realistic business process settings.

On the other hand, a group of researchers insist that time is essentially a significant resource of process management (Saven, 2004; Giaglis, 2001). Some time-relevant properties, such as the degree of parallelism (Sun et al., 2011), the race time and elapsed time (Perry et al., 1996), etc., have been widely explored. Though being specific in depicting these time-relevant properties, they mainly focus on the conclusions of their discoveries from empirical studies or domain-specific interests, instead of providing explicit methodologies about how to exactly improve these properties.

In this paper we study an approach to optimize temporal performance of workflow-based business processes that is based on our previous research of data-centric business process modeling techniques (Zhang et al., 2014; Zhang et al., 2014). In a workflow-based business process, a business activity is defined to consist of atomic activities, data types, and human actors. Assuming deterministic components of business activities that are not redesigned in our approach at this stage, our approach aims at reconstructing business processes by modifying the execution order of business activities, that can be represented by relocating business activities from their original subprocesses to others.

With an apriori knowledge (i.e., either known or estimated) of time of each business activity (Sun et al., 2011; Lanz et al., 2013), our approach is initiated by determining whether a business activity is eligible to be relocated. Given a subprocess that a business activity is not originally located in, this business activity is determined to be eligible for relocation to this subprocess only when it is, as defined, forwardly moveable, backwardly moveable, or loosely-parallelled moveable to the subprocess.

For any subprocess, given the working time that is defined as the total lapse time to complete a process unit (e.g., business activity, subprocess, and process), as in (Perry et al., 1996 and Perry et al., 1994), its waiting time is defined as the time duration of its idle state when other subprocesses are being executed. Typically, the time optimization of a subprocess is essentially to reduce its waiting time by filling its idle states with executing business activities from other subprocesses. By selecting the eligible
activities due to their contributions to reducing waiting time of the subprocess, our algorithms optimize its temporal performance by executing those activities within it.

We also develop an algorithm to approach optimal temporal performance for the overall business process that roots from the algorithm of optimizing temporal performance for a single subprocess and modifies the criteria of eligible activities to emphasize their contributions to optimizing temporal performance of the overall business processes.

Furthermore, a symbolic business process generator, namely G-DCBP (Zhang et al., 2015), is developed to generate symbolic business processes for evaluating our time-optimization approaches. Specifically, G-DCBP generates process components (i.e., data, activities, subprocesses, and clusters) and creates a hierarchy that incorporates them to build data-centric business processes. The inputs of G-DCBP are simply the parameters (mean and deterministic) of the process components that are used to derive their runtime values by applying pre-determined stochastic models.

In G-DCBP, data symbolization is a fundamental factor in demonstrating and generating data-centric business processes. Data\(^1\) are symbolized by combining English letters in different data types (e.g., input_data, output_data) defined by Zhang et al. (2014). Accordingly, activities are delineated by their associated data. Subprocesses are represented with serializing activities and clusters are represented with parallel subprocesses.

Data flow patterns, are another important factor of constructing data-centric business processes and stating the global business effects, properties, and objectives in G-DCBP. Data flows are described by transitions and transmissions of data in both macro- and micro-levels. A micro-level data flow indicates that newly-generated data, along with the data that is inherited from the previously-executed activities, are assigned to the associated unexecuted activities. Macro-level data flows are delineated by aggregating micro-level data flows.

In the G-DCBP implementations of the time-optimization algorithms, two based scenarios, light-load and heavy-load, are designed to evaluate the impact from different process components on how the temporal performance of data-centric business processes can be optimized. The results indicate that the data dependency and size of business processes can have the significant impacts on its temporal performance while other components do not.

The rest of the paper is organized as follows. Section 2 introduces related work. Section 3 describes our approach that optimizes temporal performance of data-centric business processes. Section 4 introduces G-DCBP. Section 5 uses G-DCBP to study the efficacy of the time-optimization approaches. Section 6 discusses our conclusions.

2. RELATED WORK

Scheduling is one major approach to optimize time-relevant properties in business processes. Floudas et al. (2005) present an overview of mixed integer linear programming (MILP) based approaches to schedule chemical processing systems. By representing time in a discrete and continuous manner, they develop effective models for a variety of chemical processes and efficient solutions for difficult MILP models in a short-term scheduling domain. Using the similar methods to represent time, Pinto et al. (1998) present two scheduling models: single-unit assignment models where task assignment is predetermined and multiple-unit assignment models where objects compete for processing products. Mendez et al. (2006) aim at extending scheduling techniques of batch processes to handle large volume processes as well as different objectives. However, they are argued to be limitedly applied in the scenarios in which constraints cannot be mathematically modeled. Moreover, most of the techniques are NP-complete such that closed-form solutions cannot be provided by merely deriving those mathematical models.

Time sometimes is considered as a reference for modeling techniques. Giaglis, (2001) and Mili et al. (2004) include time as the dynamic view along with informational, functional, and organizational views together as the fundamental views to construct business processes. Assuming apriori knowledge of working time of each business activity, Sun et al. (2011) present methods to detect the degree of parallelism that is defined as the number of the running subprocesses at the same time. Perry et al. (1996) and Perry et al. (1994) implement empirical studies of software developments and try to identify the factors to cause the discrepancy of race time, that is defined as the time spent on actual work and elapsed time, that additionally includes the presence of interruption, blocking, and waiting periods. However, these researches fail to provide in-depth approaches for how exactly time performance can be improved by applying their discoveries. Our approach, by analyzing the eligibility of how a business process can be optimized, provides methods to effectively optimize time performance for both a single subprocess and overall business process.

Data-centric business processes have long been an important research area. Bhattacharya et al. (2009) and Cohn et al. (2009) introduce the design methodology that focuses on business artifacts and that includes both the business-relevant data and their macro-level lifecycles. Furthermore, automatic verifications are applied to test whether runs of an artifact system satisfy desirable correctness properties (Deutsch et al., 2009). Data-centric modeling techniques have been widely used in web services,
such as (Baghdadi, 2005), that propose a business model with multiple interfaced abstraction levels to methodologically deploy web services technology with the attributes describing the business artifacts as described in the highest abstraction level of the business model. To evaluate whether technologies are appropriate for each tier of n-tier architectures in eCommerce, a data-centric design is proposed in (Manuel et al., 2003) to determine whether it is necessary to move data to communicate messages between applications.

Business process simulators/generators are one important set of tools in the research of business process modeling. Protos (Verbeek et al., 2005) is one typical application that applies Petri Net to define business processes by business process redesign and communication enhancement of process stakeholders. FileNet (Netjes et al., 2006) graphically models process structures, has them passed to the engine for execution, and aggregates and parses the data for subsequent analysis. CPN tools (Jensen et al., 2007) are one of the general business process tools that edit, simulate, and analyze Colored Petri Nets with friendly user interfaces and novel interaction mechanisms. However, they are designed with complicated input settings and few of them are used to generate data-centric business processes.

3. Approach Overview

In this section, we introduce our approach to optimize temporal performance of business processes that is modeled by our data-centric business process modeling techniques. The approach is classified to two stages: eligibility verification of business activity relocation and the algorithms of time optimization. Generally, it is initiated by identifying the business activities that are eligible to be executed in a given subprocess. With the knowledge of the working time of each eligible business activity, our algorithms select the ones to approach an optimal temporal performance for both single subprocesses and overall business process.

3.1 Data-Centric Business Activity

In our previous research (Zhang et al., 2014), a business activity being composed of a tuple of components (data, roles, and atomic activity) is formed when one or more atomic activities are executed in order by identical role(s). Data is further modeled to indicate its states in data flows within a business activity as initial data, input data, global data, consumed data, output data, and final data, with their properties simply introduced as follows:

- **Initial_data**: data that is initially injected to the data flows and triggers the execution of the associated business activity.
- **Input_data**: data that is generated and delivered by other business activities and triggers the execution of the associated business activity.
- **Global_data**: data that is injected independently from business processes and triggers the execution of the associated business activity.
- **Consumed_data**: input data or initial data that is consumed by the execution of the associated business activity.
- **Output_data**: data that is deliverable by the execution of the associated business activity.
- **Final_data**: data that is generated by the execution of a business process and not consumed by any business activity, or a state to indicate the end of the execution of a business process.

In our approach, we do not consider global_data for its contribution to evaluate any property of a business process because it is a constant.

A subprocess is defined to be composed of business activities in execution order and bounded by their convergence or divergence relations.

Subprocesses that are composed of activities and bounded by the identical convergence or divergence relations are defined as a cluster.

3.2 Eligibility for Business Activity Relocation

In our data-centric business process, a business activity is executed in a chronologically static manner when its input is delivered by its adjacent previously-executed activities, or its output is immediately absorbed by its adjacent subsequently-executed activities. On the other hand, it is possible that a business activity can be flexibly executed if it is not directly dependent on the execution of its adjacent previously-executed or subsequently-executed activities.

Our data-centric business process modeling technique specifies this possibility by defining that for any execution of adjacent business activities, the input_data of a business activity is not necessarily the output_data of its adjacent previously-executed activity. Specifically, assume a business activity, namely WA, with its input_data being part of the output_data generated only in a subprocess SP_A that is executed earlier but not adjacent ahead. Then executing WA adjacent following SP_A is legitimate (assuming no other sources of impact), because this flexible execution does not impact on their original order of data transmission that ensures the execution of the business process. Analogously, if the output_data of WA is the input_data of a subprocess SP_B and SP_B is executed not adjacent later, then the postponed execution of WA adjacent ahead of SP_B is legitimate as well. This observation provides the fundamentals for flexible execution order of business activities that can be reflected as relocating business activities using our data-centric modeling technique.
This reflection provides a possibility of deriving a partial order of any given subprocess and activity. Assume process flows can be represented, i.e., in our AST-based approach (Zhang et al., 2014) in this paper. Define TP(A) to denote the set of process components ahead of activity A, and TP(SP) to denote the set of process components ahead of subprocess SP. The partial order between A and SP can be derived by comparing the elements in TP(A) and TP(SP).

**Theorem 1.** Activity A is executed before subprocess SP when TP(SP) \( \subseteq \) TP(A), and Activity A is executed after subprocess SP when TP(A) \( \subseteq \) TP(SP).

Typically, any TP that is a subset of another indicates the corresponding activity/subprocess are in the same process flow, and their execution order can be completely identified.

**Theorem 2.** Activity A is loosely parallel to subprocess SP when TP(SP) \( \notin \) TP(A), and TP(A) \( \notin \) TP(SP).

As opposed to Theorem 1, the fact that the TPs of a pair of activity/subprocess are not mutually subsets of one another indicates disjoint process flows between them. This parallelism is considered to be loose because the chronology of A and SP can actually be detected by deriving the total order of activities/subprocesses, yet it is not necessary in our approach at this point.

In the data-centric modeling technique, a workflow-based business process is perceived as being established and maintained by its process flows, i.e., the execution of a business process is realized by the data delivery among business activities. Hence to explore the eligibility of relocating a business activity is essentially to find out whether the data delivery in the business process would be impeded after relocating the business activity. Assuming in our approach at this point that a business activity is kept consistent without modifying its components (i.e., data types, human actors, and atomic activities), data delivery is ensured to be safe when its pattern (i.e., the order of data transmission among business activities), is kept identical with that in the original process flows. Particularly, for any relocated activity, the data delivery of a business process is secured when the input_data of that activity is part of the data that is delivered by all its previously-executed activities, and the output_data of that activity is delivered to its subsequently-executed activities in its process flow.

**Definition 1.** Starget denotes a subprocess. Atarget denotes an activity that is not part of Starget. Sinbetween denotes all the subprocesses between Starget and Atarget. InputAtarget, outputAtarget, InitialAtarget, and consumedAtarget respectively denote the sets of all the input_data, output_data, initial_data, and consumed_data of Atarget. Sbeforearget denotes the set of the subprocesses that are executed before Starget (including Starget) in its process flow. InputSbeforearget, outputSbeforearget, InitialSbeforearget, and consumedSbeforearget respectively denote the sets of the input_data, output_data, initial_data, and consumed_data of all the activities of Sbeforearget. Sinbetween denotes the set of input data of Sinbetween.

Sbeforearget can be derived by searching the TPs that contain TP(Starget). This search can be simple since TPs have already been sorted by their mutual commonalities and lengths of process components.

**Theorem 3.** Atarget is backwardly moveable to Starget when 1) Atarget is executed after Starget; 2) inputAtarget only belongs to (inputSbeforearget \( \cup \) initialSbeforearget \( \cup \) outputSbeforearget) – consumedSbeforearget and none of the consumedAtarget belongs to inputSinbetween.

Theorem 3 illustrates under what circumstances the execution of Atarget is eligible to be executed adjacently following the execution of Starget. According to the data-centric modeling technique, it is possible that in Sbeforearget, a single piece of data is transmitted among different activities as multiple data types. (InputSbeforearget \( \cup \) initialSbeforearget \( \cup \) outputSbeforearget) - consumedSbeforearget ensures accurate data delivered from Sbeforearget. Since Atarget is brought forward, the impact of its output data delivery to the subsequent subprocesses stays consistent and is not necessarily cared about.

**Definition 2.** Saftertarget denotes the set of the subprocesses that are executed after Starget (including Starget) in its workflow. InputSaftertarget, outputSaftertarget, initialSaftertarget and consumedSaftertarget respectively denote the sets of the input_data, output_data, initial_data, and consumed_data of all the activities of Saftertarget.

As opposed to the way of deriving Sbeforearget, Saftertarget can be derived by searching the TPs that are contained TP(Starget).

**Theorem 4.** Atarget is forwardly moveable to Starget when 1) Atarget is executed before Starget; 2) none of outputAtarget - (outputAtarget \( \cap \) inputAtarget) belongs to inputSinbetween.

Theorem 4 illustrates under what circumstances the execution of Atarget is eligible to be postponed to be adjacently ahead of the execution of Starget. Since executing Atarget is postponed, the impact from its previously-executed subprocesses to its input data stay consistent and is not concerned about. In addition, it is necessary to make sure that the postponed execution of Atarget does not damage the process flow of Sinbetween, that can be reflected by verifying the newly generated output_data after executing Atarget is not the input_data of any subprocess of Sinbetween.
**Theorem 5.** A target is loosely-parallelled moveable to Starget when 1) A target is loosely parallel to Starget; 2) inputAtarget only belongs to (InputSbeforetarget U initialSbeforetarget U outputSbeforetarget) – consumedSbeforetarget and none of the consumedAtarget belongs to any of InputSaftertarget; 3) none of outputAtarget – (outputAtarget \ supplement inputAtarget) belongs to the input data of any of the partial subprocess that are executed after the subprocess that A target is originally located in.

Obviously, the conditions to trigger a loosely-parallelled moveable business activity are more constrained compared with those of Theorem 4 and 3.

In summary, given a subprocess and an activity, the activity is eligible for relocation in the business process when it can be forwardly moveable, backwardly moveable, or loosely-parallelled moveable to a different subprocess.

### 3.3 Algorithms of Temporal Optimization

In this paper, time inefficiency is considered to be caused only by the waiting time of subprocesses that is generated after their executions are completed while other subprocesses are still being executed. Given the apriori knowledge of the working time of each business activity of business process, the working time of a subprocess can be calculated by simply summing up the working time of all its activities. Then the waiting time of that subprocess is derived by being subtracted from the working time of its associated cluster. Note in our approach, working time and waiting time are considered as worst-case metrics. Particularly, the working time of a cluster is defined as the largest working time of all its associated subprocesses.

#### 3.3.1 Time Optimization for A Single Subprocess

It is possible in reality that only one (or a few) subprocess needs to optimize its temporal performance to achieve certain business goals. In our approach, to optimize temporal performance for a single subprocess, namely Starget is simply equivalent to reduce its waiting time by executing eligible activities from other subprocesses.

Each activity in the business process is verified beforehand whether they are eligible for Starget, that is, whether they are forwardly moveable, backwardly moveable, or loosely-parallelled moveable to Starget. There might be multiple eligible activities, namely Aeligible from other subprocesses, all of which could contribute to reducing waiting time of Starget. Ideally, they are considered to be selected and relocated together to Starget. However, it is only allowed to have one Aeligible relocated to Starget at one time, because after any activity is relocated to Starget, the patterns of data delivery of Starget need to be updated accordingly. It is possible that multiple Aeligible might be “incompatible” with each other’s process flow, such that data delivery of Starget would be impeded if they are relocated together to Starget.

Furthermore, this brings up a typical NP-hard problem. Each time when any Aeligible is relocated to Starget, it is necessary for Starget to update the eligibility of its associated activities and reselect the eligible activities accordingly. In fact, under the constraints of our approach, the optimal waiting time cannot be obtained unless all the possible choices of the Aeligible combinations are tried out and each of their performance relative to waiting time optimization is compared. Even given a solution of Aeligible combination, the only way to find out whether it is optimal is to enumerate all the possible Aeligible
combination and compare them. This can be generalized that the optimality of a given solution of this problem cannot be ensured by any method to our knowledge within polynomial time. Therefore, to optimize temporal performance for a single subprocess is an NP-hard problem under our constraints of this approach.

To approach an optimal solution in this case, techniques of approximation need to be adopted in our algorithms. Initially, the absolute value of the contribution to time optimization from each A\textit{eligible}, namely AB\textit{time}, is calculated as the waiting time of \textit{S\textit{target}}, namely WA\textit{target}, subtracting the working time of each A\textit{eligible}, namely WO\textit{eligible}. A\textit{eligibles} are sorted in an ascending order of their AB\textit{time}.

After the first element in \{A\textit{eligible}\} is extracted, \{A\textit{eligible}\}, along with \{WA\textit{eligible}\} and \{AB\textit{time}\}, are updated according to the varying pattern of the data delivery of \textit{S\textit{target}}. After sorting \{AB\textit{time}\}, the smallest one is compared with WA\textit{target}. If it is larger than WA\textit{target}, then WA\textit{target} is the optimal waiting time of \textit{S\textit{target}} and no further update is needed. Otherwise the above process repeats until a smaller WA\textit{target} is found. This algorithm, instead of enumerating all the possibilities, affirmatively selects the observed solutions to approach an optimal waiting time. Its pseudo codes are listed in Algorithm 1.

3.3.2 Time Optimization for the Overall Business Process

The algorithm to optimize temporal performance of the overall business process roots from that of a single subprocess. Instead of simply reducing the waiting time in one single subprocess, the techniques of time optimization for overall business process aims at optimizing working time of the overall business process, that is approximated by reducing the working time of a cluster in our approach (Zhang et al., 2014)

Particularly, AB\textit{time}, that is calculated simply as the absolute value of WA\textit{target} subtracting WO\textit{eligible}, cannot accurately reflect the overall time optimization. In fact, not only the contribution from WO\textit{eligible} after A\textit{eligible} being relocated to \textit{S\textit{target}}, but also the impact that the relocation of A\textit{eligible} imposes on its original subprocess needs to be considered to contribute to the varying temporal performance of the overall business process. Define VA\textit{original} as the varying working time of the cluster where A\textit{eligible} is originally located. VA\textit{original} is calculated as the working time of the original cluster before A\textit{eligible} being relocated, subtracting the working time of the original cluster after A\textit{eligible} being relocated. Similarly, VA\textit{target} denotes the varying working time of the cluster that needs to optimize temporal performance and is calculated as the working time of that cluster after A\textit{eligible} being relocated subtracting the working time of that cluster before A\textit{eligible} being relocated.

VA\textit{sum}, defined as the summation of VA\textit{original} and VA\textit{target}, is used to indicate how the relocation of A\textit{eligible} changes the working time of the original and relocated clusters. If VA\textit{sum} is negative, then the total working time of these clusters is decreased, and vice versa. For any A\textit{eligible}, the more negative VA\textit{sum} is, the more improved the temporal performance of the overall business process gains by relocating the A\textit{eligible}.
Ideally, to achieve the time optimization of overall business process, it is equivalent to realizing the time optimization of each subprocess and summing them up. However, each time after one Aeligible is relocated to a subprocess, the subprocesses whose pattern of data delivery is impacted need to update their associated {Aeligible}. Moreover, the possibility of a relocated Aeligible being selected and relocated by another subprocess at a later stage increases the uncertainty of identifying an optimal solution of reconstructing the business process. Therefore, to optimize temporal performance of the overall business process is at least as hard as that of a single subprocess and can be perceived as an NP-hard problem as well. To efficiently approximate the optimal temporal performance of the overall business process, an array, namely {Aabandon} is defined to store the Aeligible that has been relocated to make sure that even Aeligible might be optimal for other subprocesses at a later stage, they would not be relocated anymore. This process ensures the reduction of the number of the remaining Aeligibles.

Our algorithm is initiated by determining {Aeligible} for each Starget. {Aeligible} is then sorted along with {VAsum} in an ascending order. All the first elements of all the {VAsum} are then compared. The Aeligible associated with the smallest VAsum is relocated to its corresponding Starget. Then this Aeligible is inserted to {Aabandon} that indicates this Aeligible cannot be relocated by any subprocess at a later stage. This process is repeated in our algorithm.

One Starget quits relocating Aeligible when either its VAsum associated with the Aeligible is a positive value, that means the working time of the original and relocated clusters would be increased after the relocation of the Aeligible. or no Aeligible can be found.

The pseudo code of this algorithm is listed in Algorithm 2. The process of calculating VAoriginal and VAtarget is omitted here for simplicity.


To evaluate the efficacy of the time-optimization algorithms, we develop a symbolic business process generator, namely G-DCBP. G-DCBP maps the concepts of data-centric business processes introduced in our previous paper (Zhang et al., 2014) to simulation codes by: 1) establishing the hierarchy that connects and layers activity, subprocess, cluster, and process; 2) taking inputs in simple formats and outputting business processes with simulated components; 3) symbolizing data objects, assigning them to activities, and attributing them to different data types; 4) deciding data flow patterns among activities, subprocesses, and clusters; 5) providing easy access for associating the simulated business processes with the properties that need to be analyzed or evaluated.

The mechanisms of G-DCBP are presented in Fig.1.

4.1 Hierarchy

In our previous papers (Zhang et al., 2014; Zhang et al., 2014), a business activity is defined to be a tuple of data and atomic activities that represents an independent unite of work; a subprocess is defined to be composed of business activities in execution order and bounded by their convergence and divergence relations; a cluster is defined as subprocess bound by identical convergence and divergence relations. In G-DCBP, these concepts are correspondingly mapped to be the components: activities, subprocesses, clusters, and process, that are implemented class-wise in simulation codes, where subprocesses inherit activities, clusters inherit subprocesses, and process inherit clusters. That indicates that the derived components in the hierarchy can obtain the shared properties from their super components, from which their corresponding properties are aggregated or deduced.

4.1.1 Activity

An activity is the fundamental component of the hierarchy where the data types (e.g., input_data, output_data, etc.) and their associated settings are defined.

4.1.2 Subprocess

A subprocess applies stochastic models (e.g., uniform, Gaussian, etc.) to determine the runtime data number of each subprocess and assign the corresponding data to each of its associated activities. Micro-level data flows among activities within the identical subprocess are realized by a dataset structure, namely inherit_data_pool that collects the data objects which can be delivered to the subsequently-executed activities as input_data.

4.1.3 Cluster

A cluster applies stochastic models to determine its associated subprocess number of each cluster and the corresponding subprocess, where the macro-level data flows are realized by cluster-wise inherit_data_pool aggregating the subprocess-wise ones to deliver data to the subsequently-executed clusters.

4.1.4 Process

Process is composed of clusters and configured by their placements. A cluster can be categorized as beginning_cluster, join_cluster, split_cluster, join_and_split_cluster, or singular_cluster, due to the
number of the clusters that triggers it and/or the number of the clusters that its execution triggers. **Beginning_cluster** denotes that the execution of a cluster is triggered by no other clusters and triggers the execution of another cluster. **Join_cluster** indicates that the execution of a cluster is triggered by the execution of multiple clusters and triggers the execution of another cluster. **Split_cluster** indicates that the execution of a cluster is triggered by the execution of another cluster and triggers the execution of multiple clusters. **Join_and_split_cluster** indicates that the execution of a cluster is triggered by and triggers the execution of multiple clusters. **Singular_cluster** indicates the execution of a cluster is triggered by and triggers the execution of another cluster. Some examples of **join_cluster**, **split_cluster**, and **join_and_split_cluster** are listed in Fig. 2 for better illustration.

![Fig. 1: The G-DCBP class diagram](image)

**(a) An example of join_cluster**

**(b) An example of split_cluster**

Fig. 2: Examples of illustrating **join_cluster**, **split_cluster**, and **join_and_split_cluster**, that are represented by the black dots.

### 4.2 System I/O

G-DCBP aims at offering end users easy access for customizing data-centric business processes according to their expectations of data and the configuration of activities, subprocesses, and clusters. The detailed input parameters are listed as follows:
• **total_cluster_number**: the number of the total clusters that are needed in the expected business processes
• **beginning_cluster_number**: the number of the clusters that indicates the beginning of the execution of business processes
• **singular_cluster_number**: the number of the clusters whose execution is triggered by and triggers one cluster
• **connecting_cluster_number**: the summed number of the join_cluster, split_cluster, and join_and_split_cluster
• **mean_number_of_subprocesses_in_cluster**: the mean number of subprocesses in one cluster from which the runtime number of subprocesses in each cluster, namely number_of_subprocesses_in_cluster is derived in accordance with the stochastic models.
• **mean_number_of_activities_in_subprocess**: the mean number of activities in one subprocess from which the runtime number of activities in each subprocess, namely number_of_activities_in_subprocess is derived in accordance with the stochastic models.
• **mean_data_dependency_rate**: the mean value of data dependency rate that is defined as the identical data shared between two subprocesses divided by the total number of the data objects in each associated subprocess; the runtime data_dependency_rate is derived by this mean and the stochastic models.
• **mean_data_number_per_cluster**: the mean number of data in one cluster from which the runtime number of data in each cluster, namely data_number_per_cluster is derived in accordance with the stochastic models.

Given the deterministic input parameters regarding clusters, the mean input parameters of data, activities, and subprocesses, and the pre-determined stochastic models, G-DCBP delivers data-centric business processes that meet end users’ expectations under such simple input formats.

### 4.3 Data Symbolization

In G-DCBP, data are symbolized as the combination of English letters, e.g., “AE”, with its digits determined by number_of_data_per_cluster, data_dependency_rate, and the data that have already been designed. Particularly, data is generated per cluster and assigned to the underlying subprocess.

The digit number of data symbolization can be calculated as equation (1), where \( n \) denotes number_of_data_per_cluster of the associated cluster, \( D_i \) denotes the number of data in the \( i \)th cluster, and \( m \) denotes the number of the clusters that have been generated. An example is used to illustrate the mechanism. Assume 100 pieces of data are generated for a cluster \( c \). The number of the assigned data for the generated cluster is 2000. It can be derived that this scenario needs to be applied with the third sub-equation of equation (1). Therefore, the digit number of that data symbolization in \( c \) is \( \text{ceil}(\log_{26}2000) \), which is 3.

During the phase of generating one subprocess \( s \), the number of the data assigned from the generated data of the associated cluster \( c \), namely number_of_data_in_subprocess is calculated as \( \text{data_number_per_cluster}_c \times \text{data_dependency_rate} \). The corresponding data are randomly determined from the generated data in \( c \) afterwards. The generation of these data follows the alphabetic order.

The entire process above is iterated to assign data for every subprocess.

#### 4.4 Data Flow Patterns

Data-centric business processes depict the overall effects of business processes by data flows that are represented as the transitions and transmissions between data types, in both macro- and micro-levels.

To establish micro-level data flows, after a subprocess is created with its number_of_activities_in_subprocess, its associated number_of_data_in_subprocess, and the corresponding data are subsequently derived. G-DCBP partitions these data to the activities and assigns them as different data types (e.g., input_data, consumed_data).

Typically, a piece of data is either consumed by the execution of its activity or delivered to the subsequent activities for their executions. However, flow patterns can be complicated when data is not instantaneously delivered to the subsequent activities for their executions. These data are collected by Inherit_data_pool that is applied both subprocess-wise and cluster-wise. A subprocess-wise Inherit_data_pool collects the input data and the inject data that has not been consumed, and the output data each time when an activity is iterated during the phase of generating subprocesses. For each activity, its input_data is determined from its associated Inherit_data_pool. This process of updating Inherit_data_pool and selecting the input_data is iterated for every activity to enable the micro-level data flows. Similar to subprocess-wise Inherit_data_pools, cluster-wise Inherit_data_pools collect subprocess-wise Inherit_data_pools within the same clusters.
Data flow patterns can be illustrated by examples. Assume 2 activities, labeled 1 and 2, and 5 pieces of data, labeled A to E, are assigned to a subprocess. In a G-DCBP implementation, activity 1 is assigned with data A, B, C, where A and B are the inject_data, C is the output_data, and A is consumed by the execution. The subprocess-wise Inherit_data_pool collects B and C as the possible input data delivered to activity 2, that takes B as its input data being consumed and D as its inject_data, with E as its output_data. The instantaneous subprocess-wise inherit_data_pool then is updated with C, D and E, that are delivered to the subsequently executed subprocesses.

Macro-level data flows are realized by cluster-wise Inherit_data_pools, that are formed by aggregating data from all its associated subprocess-wise Inherit_data_pools. Moreover, since the neighboring clusters that is executed before (known as parent_cluster) and after (known as child_cluster), the current clusters are identified in the cluster component of G-DCBP, inter-cluster data delivery can be realized by simply acquiring the Inherit_data_pool form the parent_cluster and delivering its own Inherit_data_pool to the child_cluster, thus ensuring the macro-level data flows.

4.5 Easy Access

G-DCBP delivers business processes with straightforward hierarchy of concepts, i.e., activities, subprocesses, and clusters that can be easily accessed by simply referencing their classes and creating the corresponding objects, fields, and functions. Moreover, the process module collects all the activities, subprocesses, and clusters for easy access by end users.

5. Evaluation of Temporal Optimization Approaches

In this section, G-DCBP is used to implement our time-optimization approach and evaluate its efficacy. According to the preset input parameters, G-DCBP generates symbolic business process that delivers its associated components, i.e., clusters, subprocesses, and activities. The implementation includes two modules: eligibility verification for business activity relocation and the algorithm to optimize temporal performances.

The module of eligibility verification for business activity relocation takes all the components of the symbolic business process, applies the principles of the eligibility for business activity relocation, and delivers the sets of subprocesses with their associated eligible activities.

The module of the algorithm to optimize temporal performances collects the eligible activities, calculates and sorts their respective temporal gains for the subprocesses that they can be relocated to, and relocates the activities to their corresponding subprocesses according to their rankings for an optimal overall temporal performance of the business processes.

5.1 Time Assignments

In our G-DCBP implementations, working time is randomly generated and assigned to each activity. Therefore, the working time of a subprocess can be derived by summing the working time of its associated activities, and the working time of a cluster is calculated as the maximum working time of its associated subprocesses.

5.2 Temporal Gain Calculations and Rankings

In this paper, our optimized business goal is the working time of the overall process. Therefore, the temporal gain of moving an activity is defined as the difference between the temporal gain of its originally associated cluster after being moved from, namely $G_{cf}$, and the temporal gain of its newly associated cluster after being moved to, namely $G_{ct}$. Specifically, the temporal gains for moving an activity is calculated as follows:

$$temporal\_gain = G_{cf} - G_{ct}$$ (2)
with \(G_{cf}\) and \(G_{ct}\) calculated as equation (3) and (4), where \(WTA\) denotes the working time of an activity \(A\), \(WT_{sf}\) denotes the working time of \(A\)'s originally associated subprocess where \(A\) is relocated from, and \(WT_{cf}\) denotes the working time of \(A\)'s originally associated cluster where \(A\) is relocated from. \(max\{WTA\}\) denotes the updated maximum working time among all the subprocesses within the cluster where \(A\) is originally associated with after \(A\) being relocated, \(WT_{st}\) denotes the working time of \(A\)'s newly associated subprocess where \(A\) is relocated to, and \(WT_{ct}\) denotes the working time of \(A\)'s newly associated cluster where \(A\) is relocated to.

To record the subprocesses that each activity can be relocated to and the corresponding temporal gains, a lookup table, namely \(activity\_gain\_table\) is designed for each activity, with the subprocesses as keys and the corresponding gains as values. Furthermore, all the temporal gains from all the activities are sorted in a list, namely \(activity\_gain\_rankings\) in a descending order.

The temporal performance improvements start off by obtaining the top element in \(activity\_gain\_rankings\) and comparing it with each activity's \(activity\_gain\_table\). After matching the value of the entry in the \(activity\_gain\_table\), the associated activity \(A\) can be found and the corresponding subprocess \(S\) can be identified as the subprocess that is expected to improve the temporal performance of the business processes after \(A\) being relocated to it. \(A\) is then relocated to \(S\) with the relevant information (e.g., \(activity\_gain\_rankings\)) being updated.

Subsequently, \(A\) is marked as unmoveable that indicates it would not be relocated to other subprocesses even when temporal performance of business processes can be improved, as explained in by Zhang et al. (2014). \(A\)'s corresponding entry in \(activity\_gain\_table\) is deleted accordingly. This process repeats until all the eligible activities are iterated and relocated to their respective subprocesses before an optimal temporal performance of business processes is approached.

5.3 Evaluations

5.3.1 Scenarios

Two base scenarios are adopted to evaluate our approach to improve the temporal performance of data-centric business processes, with the uniform distribution:

- light-load scenario:
  - total_cluster_number: 10

The evaluations are conducted by adjusting the parameters upon the base settings.

5.3.2 Metrics

- \(temporal\_improvement\_ratio\): calculated as the total working time of business processes after being applied with the time-optimization algorithm divided by the total working time of business processes before being applied with it
- \(occurrence\_ratio\): defined as the number of simulation runs where the temporal performance improvements occur after being applied with the algorithm to optimize temporal performance divided by the total number of simulation runs in G-DCBP.

5.3.3 Results

To better understand the causality of the metrics and the process components, the evaluation is conducted with multiple groups of \(mean\_number\_of\_activities\_in\_subprocess\) (i.e., 5, 10, and 15 average activities per subprocess).

5.3.3.1 Light-load scenario vs. mean_activity_working_time
Fig. 3(a) shows the temporal improvement ratio in terms of the mean activity working time ranging from 5 to 25 of the light-load scenarios. It can be observed that the temporal improvement ratio does not vary significantly with the updates of the mean activity working time (in general between 1.5% and 3.5%).

What is more, though the subprocesses with more average activity number tend to have better temporal gains, they are not significant.

Fig. 3(b) illustrates the occurrence ratio in terms of the varying mean activity working time (in general 37% to 75%). Similar to the temporal improvement ratio performance, the occurrence ratio of temporal improvement is not significantly correlated with the mean activity working time and mean_number_of_activities_in_subprocess.

5.3.3.2 light-load scenario vs. mean_data_number_per_cluster

Fig. 4(a) illustrates the temporal improvement ratio in terms of mean_data_number_per_cluster. It can be observed that temporal improvement ratio is not significantly correlated with either mean_number_of_activities_in_subprocess or mean_data_number_per_cluster.

It can be speculated that under light-loaded scenarios, the temporal performance of business processes does not have much to do with parameters of minor process components.

Fig. 5(a) shows the temporal improvement ratio in terms of mean activity working time in the heavy-load scenarios. Similar to the light-load scenario, it can be observed that the temporal improvement ratio is not significantly correlated with either mean activity working time or mean_number_of_activities_in_subprocess. On the other hand, compared with the temporal improvement ratio in the light-load scenarios, the temporal improvement ratio in heavy-load scenario is significantly higher (15% to 23% compared with 1.5% to 3.5%).

Fig 5(b) depicts the occurrence ratio in terms of mean_activity_working_time in the heavy-load scenarios. Obviously, under the settings of the heavy-load scenario,
the temporal improvement almost occurs in every configuration of data-centric business processes.

5.3.3.4 heavy-load scenario vs. mean_data_number_per_cluster

Fig. 6(a) illustrates the temporal improvement ratio in terms of mean_data_number_per_cluster in heavy-load scenario. It can be observed that the temporal improvement ratio does not vary by too much among different mean_number_of_activities in subprocess. Moreover, the results imply a pattern that by increasing mean_data_number_per_cluster, the temporal improvement ratio tends to be lower.

Fig. 6(b) indicates that under the settings of the heavy-load scenario, the temporal improvement occurs in every configuration of data-centric business process, no matter how many data items are distributed to each subprocess.

Compared with the temporal performances in the light-load scenario, the temporal performance in the heavy-load scenario are better in terms of both temporal improvement ratio and occurrence ratio.
5.3.3.5 heavy-load scenario of 50% mean data dependence rate vs. mean data number per cluster

In real world, processes with high data dependency, e.g., repeated business services for different customers, can be limited. In this section, we investigate the effect on the temporal improvement from data dependency between subprocesses. Specifically, the mean data dependence rate is set to be 50%.

Fig. 7(a) illustrates the temporal improvement ratio in terms of mean number of data in subprocess. Compared with the heavy-load scenario with 100% mean data dependence rate, the temporal improvement ratio of the heavy-load scenario with 50% mean data dependence rate is significantly weaker. For instance, for the processes with mean activity number of 10, the temporal improvement ratio of the heavy-load scenario with 100% mean data dependency rate is around 13% to 25%, while it is only around 1% to 4% in the heavy-load scenario with 50% mean data dependency rate. In addition, it also can be observed that the processes with more mean number of activities in subprocess tend to have better temporal improvement ratio performance (0.1% to 1% of 5 mean number of activities in subprocess and 1% to 4% of 10 mean number of activities in subprocess). On the other hand, by increasing mean data number per cluster, the temporal improvement ratio significantly decreases when mean data dependency rate is 50%.

Fig. 7(b) demonstrates the occurrence ratio in terms of mean data number per cluster. It can be observed that the processes with fewer mean number of activities in subprocess have a lower occurrence ratio than the processes with more.

5.3.4 Analysis of Evaluation Results

The evaluation results deliver the different impacts from the process components as follows:

- The performance of temporal optimization of data-centric business processes mainly relies on how subprocesses are correlated with each other with respect to their data dependencies. The higher data dependencies are, the more chances business activities can be relocated and the better the temporal improvement can be.
- Under the same level of data dependencies, the temporal improvement performance is mainly subject to the size of the process. Typically, large-size data-centric business process tends to have better temporal improvement performance than the small-size ones.
- Besides data dependencies and the size of business processes, the number of data items of a subprocess can affect the temporal improvement of data-centric business processes. This affect is more significant for large-size business processes. Typically, in large-size data-centric business processes, the temporal improvement ratio is lower when the number of data items of each subprocess is larger.
- The parameters relevant to minor process components (e.g., activities) do not impact significantly on the temporal improvement performance of data-centric business processes.

6. Conclusions

In this paper we study a data-centric approach to optimize temporal performance in data-centric business process. Assuming deterministic business activities whose components stay consistent, we proposed approaches of reconstructing the original business process by relocating the business activities in the business process. In Our approaches, a business activity can be relocated to a given subprocess if it can be forwardly moveable, backwardly moveable, or loosely-parallelled moveable to a subprocess.

Furthermore, we explore how temporal performance of a single subprocess can be optimized. It is observed that time inefficiency of one subprocess is caused when its execution is completed while others are still being executed. By collecting the eligible business activities, we develop algorithms to relocate them to that subprocess to reduce its waiting time. Moreover, we develop an algorithm to explore how to optimize temporal performance of the overall business process. Based on the methods to optimize time performance of a given subprocess, we approach optimal temporal performance of the overall business process by modifying the criteria of eligible activities.

We further introduce a tool that generates symbolic data-centric business processes with stochastic models, namely G-DCBP. Its major advantages are: 1) it adopts simple for- mats of inputs and generates symbolic data-centric business processes according to the end users’ specifications. 2) all the symbolic process components (i.e., activities, subprocesses, clusters, and processes) are provided and easily referenced.

G-DCBP is designed as a straightforward hierarchy with its components mapped from our data-centric modeling techniques. By adopting inputs in simple formats in both deterministic and stochastic manners, G-DCBP starts to build business processes by deriving the representations of data according to the preset data number. To realize the micro-level data flows, a shared structure is implemented to collect the data that is delivered by the previously-executed business activities and pass them for the execution of the subsequent business activities. Furthermore, macro-level data flows are realized by aggregating micro-level data flows.

G-DCBP is used to evaluate the time optimization approaches. According to the results, how data are
correlated with each other impacts the most for the performance: the more data are correlated, the more the temporal performance is improved. The temporal performance can also be significantly impacted by size of business processes and the data number, while it does not have too much to do with minor factors, such as activity-relevant factors.

7. REFERENCES


Kurt Jensen, Lars Michael Kristensen, and Lisa Wells. (2007). Coloured petri nets and cpn tools for modelling and validation of

AUTHORS

Yuqun Zhang is a PhD candidate in the Center of Advanced Research in Software Engineering (ARiSE), Department of Electrical and Computer Engineering at the University of Texas at Austin. His interests include services computing and mobile computing. Yuqun received his MS degree from the University of Rochester, USA and BS degree from Tianjin University, China.

Dr. Dewayne E. Perry is a professor of electrical and computer engineering and holds the Motorola Regents Chair of Software Engineering at the University of Texas at Austin. His research interests include software architecture, empirical studies in software engineering, software system analysis, and software evolution. He holds PhD degree from Stevens Institute of Technology.