Memetic Algorithms for Mining Change Logs in Process Choreographies[*]

Walid Fdhila, Stefanie Rinderle-Ma, and Conrad Indiono

University of Vienna, Faculty of Computer Science, Vienna, Austria
{Walid.Fdhila,Stefanie.Rinderle-Ma,Conrad.Indiono}@univie.ac.at

Abstract. The propagation and management of changes in process choreographies has been recently addressed as crucial challenge by several approaches. A change rarely confines itself to a single change, but triggers other changes in different partner processes. Specifically, it has been stated that with an increasing number of partner processes, the risk for transitive propagations and costly negotiations increases as well. In this context, utilizing past change events to learn and analyze the propagation behavior over process choreographies will help avoiding significant costs related to unsuccessful propagations and negotiation failures, of further change requests. This paper aims at the posteriori analysis of change requests in process choreographies by the provision of mining algorithms based on change logs. In particular, a novel implementation of the memetic mining algorithm for change logs, with the appropriate heuristics is presented. The results of the memetic mining algorithm are compared with the results of the actual propagation of the analyzed change events.

Keywords: Change Mining, Process Choreographies, Memetic Mining, Process Mining.

1 Introduction

As a result of easier and faster iterations during the design process and at runtime, the management of business process changes, their propagation and their impacts are likely to become increasingly important [1]. Companies with a higher amount of critical changes list change propagation as the second most frequent objective. In particular, in around 50% of the critical changes, the change necessity stems from change propagation. Thus critical changes are tightly connected to change propagation in terms of cause and effects [2].

In practice, companies still struggle to assess the scope of a given change. This is mainly because a change initiated in one process partner can create knock-on changes to others that are not directly connected. Failures of change propagations can become extremely expensive as they are mostly accompanied by costly negotiations. Therefore, through accurate assessments of change impact, changes

[*] The work presented in this paper has been funded by the Austrian Science Fund (FWF):I743.

X. Franch et al. (Eds.): ICSOC 2014, LNCS 8831, pp. 47–62, 2014.

not providing any net benefit can be avoided. Resource requirements and lead times can be accounted for when planning the redesign process [3]. With early consideration of derived costs and by preventing change propagation, bears the potential to avoid and reduce both average and critical changes [2].

Hence it is crucial to analyze propagation behavior, particularly, transitive propagation over several partners. Note that change propagation might even be cyclic, i.e., the propagation affects either the change initiator again or one of the already affected partners. This is mainly due to transitivity; e.g., when a change propagation to a partner not only results in direct changes he has to apply, but also leading to redesigns in different parts of his process. In turn, this may have consequences on the change initiator or a different partner.

This paper is based on change event logs and uses mining techniques to understand and manage change propagation, and assess how changes propagate between process partners that are not directly connected (cf. Figure 1). A novel contribution is the implementation of a memetic mining algorithm coupled with the appropriate heuristics, that enables the mining of prediction models on change event logs, i.e., no information about the propagation between partners is provided.

In the following, Section 2 illustrates a motivating example, while Section 3 presents change log formats and gives the global overview of the problem. Section 4 follows up with a set of heuristics for change mining. Based on these heuristics, we introduce a memetic change mining algorithm in Section 5, which we discuss and evaluate in Section 6. In Section 7 we discuss related work and conclude in Section 8.

2 Motivating Example and Preliminaries

A process choreography is defined as a set of business partners collaborating together to achieve a common goal. Based on [4], we adopt a simplified definition of a process choreography $C := (\mathit{\Pi}, \mathcal{R})$ with $\mathit{\Pi} = \{\pi_i\}^{i \in \mathcal{P}}$ denoting the set of all processes distributed over a set of partners $\mathcal{P}$ and $\mathcal{R}$ as a binary function that returns the set of interactions between pairs of partner; e.g., in terms of message exchanges. Typical change operations comprise, for example, adding or removing a set of activities from a process model or modifying the interaction dependencies between a set of partners. A change operation is described by a tuple (δ, π) where $\delta \in \{\text{Insert, Delete, Replace}\}$ is the change operation to be performed on the partner process model π that transforms the original model π in a new model π' [4].

Consider the choreography process scenario as sketched in Figure 1 consisting of four partners `Acquirer`, `Airline`, `Traveler`, and `TravelAgency`. In this paper, we abstract from the notions private and public processes and assume that logs with change information on all partners exist (e.g. anonymized and collected). The `Acquirer` initiates a change of its process (δ, Acq) that requires a propagation to the direct partner `Airline`. In order to keep the interaction between `Acquirer` and `Airline` correct and consistent, the `Airline` has to react on the change by inserting a new fragment F3 into its process. This insertion, in turn, necessitates a

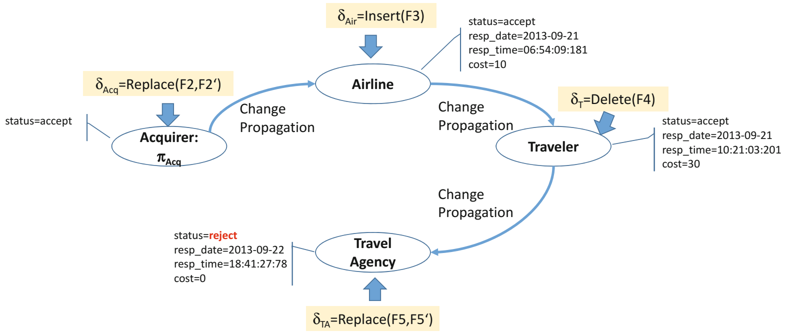


Fig. 1. Running Example: process choreography with Change Propagation

propagation to the `Traveler` that reacts by deleting process fragment F4. Finally, the change propagates to the `TravelAgency` that would have to replace fragment F5 by new fragment F5'. However, as the `TravelAgency` rejects the change, the entire change propagation fails. According to [4], such change propagation is defined as a function $\gamma : \{\text{Insert, Delete, Replace}\} \times \Pi \mapsto 2^{\{Insert,Delete,Replace\}\times \Pi}$ with $\gamma((\delta_i, \pi_i)) = \{(\delta_j, \pi_j)\}$. γ takes as an input an initial change on a given process model and generates the ripple effects on the different partners affected by the change.

The approach presented in this paper is based on change event logs collected from different partners. Figure 2 outlines the overall approach and distinguishes this work from previous ones. In [4], the overall picture of change propagation in process choreographies has been set out. Also the basic algorithms for change propagation are provided in [4]. We started analyzing change impacts in process choreography scenarios using a priori techniques in [5]. The latter work is based on the choreography structure only and does not consider information on previous change propagations that occurred between the partners.

3 Problem Formulation

In this section, we introduce two different change log types and give a global view on our approach.

3.1 Change Logs in Process Choreographies

Change logs are a common way to record information on change operations applied during process design and runtime for several reasons such as recovery and compact process instance representation [6]. For process orchestrations, change logs have been also used as basis for change mining in business processes in order to support users in defining future change operations [7].

Change logs can be also used for process choreographies. Here, every change log contains all individual change requests performed by every partner, where no propagation information are described in the log. At a certain time, all the public parts of the change logs owned by the partners participating in the collaboration

Table 1. Change Event Record

Attribute	Value
Initial Change ID	15d6b27b
Request time	2014-08-03T00:41:15
Change type	Insert
Partner	TravelAgency
Magnitude	0.6
Status	completed
Response time	2014-08-05T12:32

Table 2. Change Propagation Record

Attribute	Value
Initial Change ID	15d6b27b
Request time	2014-08-03T00:41:15
Change type	Insert
Partner	TravelAgency
Partner target	Airline
Derived change ID	c25b8c67a
Derived change type	Insert
Magnitude	0.6
Status	completed
Response time	2014-08-05T12:32

are anonymized [8], normalized, collected and put in one file to be mined. In the following, we refer to this type of log as CEL (Change Event Log).

In practice, it is also possible to have a change propagation log CPL (i.e. containing the change requests, their impacts and the propagation information as well). However, since the processes are distributed, it is not always possible for a partner to track the complete propagation results of his change requests (due to transitivity and privacy). To be more generic, we adopt change logs that contain solely change events CEL (without information about propagations) to be mined. However, in order to validate our mining approach, and assess the quality of the mined model from the CEL, we also maintain a log of the actual propagations CPL. The results of the predicted model from the CEL are compared and replayed on the CPL propagation events.

Anonymization of the logs represents an important *privacy* step [8], which is a trivial operation in a non-distributed setting. In a distributed environment a consistent anonymization scheme needs to be employed, where for example π_2 is consistently labeled as X.

Table 1 describes a sample of a change record. Each record includes information about the partner that implemented the change (anonymized), the change ID and type, the timestamps and the magnitude of the change. The latter is calculated using the number of affected nodes (in the process model), the costs (generated randomly), and the response time. Other costs can be added as needed. Table 2 describes a propagation record, with more propagation information.

3.2 Overview

As aforementioned, the main problem is to generate and analyze a propagation model by mining the change event log CEL, which contains all change events that occurred on the process partners involved in the choreography. Figure 2 gives a global overview of the main components for managing changes in collaborative processes. The first set of components (C^{3}Pro framework) provides support for specifying, propagating, negotiating, and implementing changes in choreographies. In particular, the change propagation component calculates the ripple effects of an initial change on the affected partners and checks the soundness of the collaboration if changes are accepted and implemented. The details

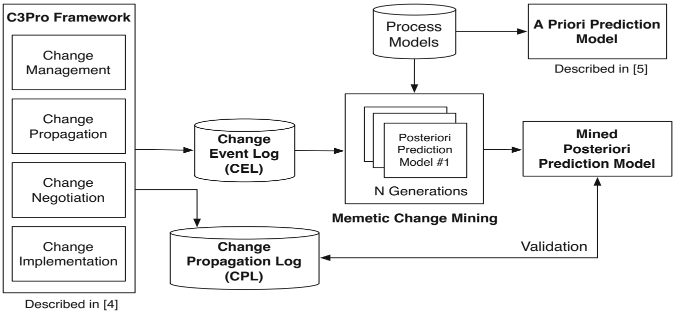


Fig. 2. Overview of the Approach

of the propagations are stored in the CPL, and all individual change events are stored in the CEL. Based on the change simulation data, posteriori and a priori techniques are provided to evaluate and understand the propagation behavior in the choreography through prediction models. The a priori technique [5] uses the choreography structure to assess and predict the impact of a change request. The posteriori technique, described in this paper, generates prediction models by mining the previously stored change data. Derived models are validated through a replay of the CPL.

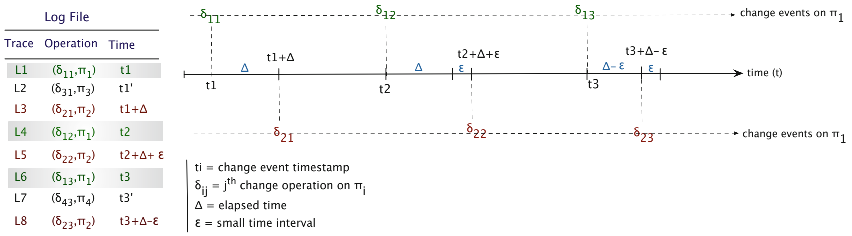
In the CEL, the relationships between the change events are not explicitly represented. The changes are collected from different partners without any information if a change on a business partner is a consequence of a change on another business partner or if they are independent (because of the transitivity). In order to correlate between the change events and understand the propagation behavior, we adopted different heuristics related to change in process choreographies.

4 Heuristics

In this section we present 4 groups of heuristics that can be exploited for mining change events in process choreographies.

Time Related Heuristic (TRH): In connection with process mining [9,10,11], if two activities a and b whose most occurrences in the log are such as the completion time of a always precedes the start time for the execution of b, then we conclude that a precedes b in the process model. If there exist cases where the execution start time of b occurs before the completion of a, then we can say that a and b could be in parallel. In change mining, a partner π_2 that always performs changes directly after a partner π_1 has changed its process, may lead to the conclusion that the changes on π_2 are the consequences of the changes of π_1. However, this does not always hold true. Indeed, the change events are collected and merged from different sources, and several independent change requests can be implemented by different partners at the same time.

In addition, in process execution logs, each trace represents a sequence of events that refer to an execution case (i.e. an instance of the process execution).



Log File

Trace	Operation	Time
L1	(δ_{11},π_1)	t1
L2	(δ_{31},π_3)	t1'
L3	(δ_{21},π_2)	t1+Δ
L4	(δ_{12},π_1)	t2
L5	(δ_{22},π_2)	t2+Δ+ ε
L6	(δ_{13},π_1)	t3
L7	(δ_{43},π_4)	t3'
L8	(δ_{23},π_2)	t3+Δ-ε

Fig. 3. Change Event Log (CEL): Representation over Time

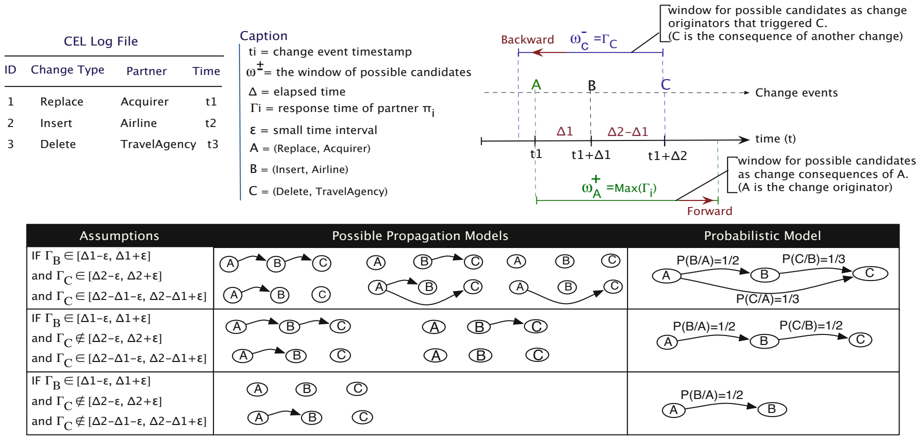
The precedence relationships between events of a same trace are explicit. In change logs CEL, each trace represents solely one change event. Even if the timestamps give an explicit precedence relationship between different traces, it is not possible to directly conclude that they are correlated. For example, we assume that the actual propagation of two initial change requests on different partners (δ_1, π_1) and (δ_2, π_2) are such that:

- the actual propagation of (δ_1, π_1) results in (δ_3, π_3).
- the actual propagation of (δ_2, π_2) results in (δ_4, π_4).

Since, in this paper, we consider that we do not have the information about the propagations, and that each of these change events is logged separately by the partner which implemented it, then, according to the timestamps, the merging and ordering of these change events in the CEL may lead to the following sequence: $[(\delta_1, \pi_1), (\delta_2, \pi_2), (\delta_3, \pi_3), (\delta_4, \pi_4)]$. According to this ordering, (δ_2, π_2) may be considered as a consequence of (δ_1, π_1) and (δ_4, π_4) as a consequence of (δ_1, π_1). In order to avoid such erroneous interpretations of the log, we need to enhance the heuristic with new elements.

Figure 3 illustrates an example of a sample CEL and its representation over time. The Figure shows a log file containing traces of 8 change events occurred on process partners π_1, π_2, π_3 and π_4 at different times. We assume that the log is chronologically ordered. According to the timestamps, there is a strict precedence relationship between the change occurrences on π_1 and the change occurrences on π_2. However, we can not directly deduce that the latter are the effects of the changes on π_1. Therefore, it is necessary to find another correlation between the timestamps that is more relevant regarding the identification of the propagation patterns. In this sense, we can remark that each time a change operation δ_1 occurs on π_1 at time t, there is a change operation δ_2 that occurs on π_2 at $t + \Delta$ with a variance of $\pm\epsilon$. This deduction holds true when the number of change occurrences on π_2 in the interval $[t + \Delta - \epsilon, t + \Delta + \epsilon]$ becomes high, and when Δ corresponds to the average latency between partners π_1 and π_2. The identification of Δ and ϵ are calculated empirically and should consider the noise in the event logs (e.g. rare and infrequent behavior).

Window Related Heuristic (WRH): Figure 4 presents another example of change events extracted from a CEL. For instance, we consider a Replace on partner `Acquirer` (A) followed by an Insert on `Airline` (B), which in turn, followed by a Delete on the `TravelAgency` (C). We also consider Γ_i as the



CEL Log File

ID	Change Type	Partner	Time
1	Replace	Acquirer	t1
2	Insert	Airline	t2
3	Delete	TravelAgency	t3

Caption
ti = change event timestamp
ω± = the window of possible candidates
Δ = elapsed time
Γi = response time of partner πi
ε = small time interval
A = (Replace, Acquirer)
B = (Insert, Airline)
C = (Delete, TravelAgency)
Backward
ω−C =ΓC
window for possible candidates as change originators that triggered C. (C is the consequence of another change)
A
B
C
Change events
Δ1
Δ2−Δ1
time (t)
t1
t1+Δ1
t1+Δ2
ω+A =Max(Γi)
window for possible candidates as change consequences of A. (A is the change originator)
Forward
Assumptions
Possible Propagation Models
Probabilistic Model
IF ΓB ∈ [Δ1−ε, Δ1+ε] and ΓC ∈ [Δ2−ε, Δ2+ε] and ΓC ∈ [Δ2−Δ1−ε, Δ2−Δ1+ε]
P(B/A)=1/2
P(C/B)=1/3
P(C/A)=1/3
IF ΓB ∈ [Δ1−ε, Δ1+ε] and ΓC ∉ [Δ2−ε, Δ2+ε] and ΓC ∈ [Δ2−Δ1−ε, Δ2−Δ1+ε]
P(B/A)=1/2
P(C/B)=1/2
IF ΓB ∈ [Δ1−ε, Δ1+ε] and ΓC ∉ [Δ2−ε, Δ2+ε] and ΓC ∉ [Δ2−Δ1−ε, Δ2−Δ1+ε]
P(B/A)=1/2

Fig. 4. Correlating change events using forward and backward windows

response time of partner i (the **average** time required by partner i to implement a change). When mining this log, the first challenge is to know if a change event is the originator or a consequence of another change. For this purpose, we define two types of windows; i.e., backward and forward. Given a change event, the forward window represents all the following change events that can be considered as effects of its propagation. In contrast, a backward window includes all previous change events that can be considered as the originators of the change event in question. For instance, in Figure 4, the forward window of A (i.e., ω_A^+), is defined by the maximum of the response times of all change events $(i.e., Max_{i \in \mathcal{P}}(\Gamma_i))$. Indeed, Γ_i is the time required by a partner to react and implement a change. So, with respect to A, if a following change event B was implemented at a time t_B such as $t_B - t_A > \Gamma_B$, then B cannot be considered as consequence of A. In turn, if $t_B - t_A < \Gamma_B$, then B might be a consequence of A (but not necessarily).

For a given change A, since we know that only change events that respect this constraint can be considered, then the possible candidate events as consequences of A should be within this window $\omega_A^+ = Max_{i \in \mathcal{P}}(\Gamma_i)$. According to this approach, in Figure 4, the possible change events that can be considered as effects of A are B and C.

This forward window allows to avoid parsing all change events that come after A to the end of the log CEL. However, a change event within this window does not necessarily imply that it is a consequence of A. for example, C is within the ω_A^+ window, but Γ_C **can be** such as $\Gamma_C < t_C - t_A < \omega_A^+$ or $t_C - t_A < \Gamma_C < \omega_A^+$. For instance, if we assume that in time scale, $t_A = 3$, $\omega_A^+ = 10$, $t_C = 9$ and $\Gamma_c = 4$, then $\Gamma_C = 4 < t_C - t_A = 6 < \omega_A^+ = 10$. In this case, C is within the window of A, but did not occur within its response time $t_A + \Gamma_C \pm \epsilon$ (ϵ is a variance value).

On the other hand, for a given change event C, the backward window ω_C^- includes all possible change events that can be considered as the originators that triggered C. In this sense, if C is a consequence of another change A, then $t_C - t_A$

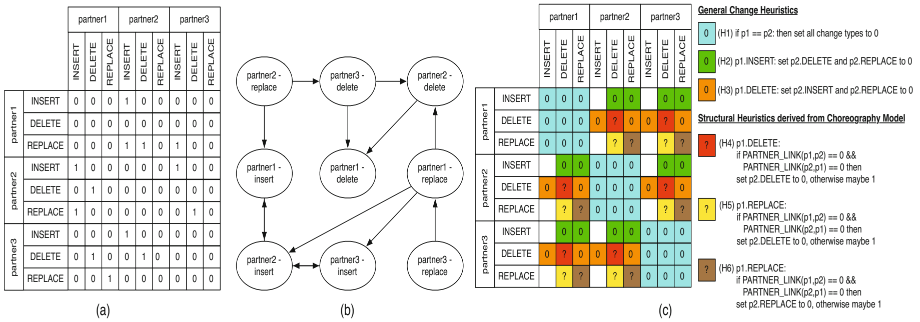
should be approximately equal to Γ_C, and therefore $\omega_C^- = \Gamma_C$. However, a change event A that occurred at $t_C - \Gamma_C \pm \epsilon$ does not necessarily mean that C is consequence of A. Indeed, both events can be independent.

Back to Figure 4, the table shows the possible propagation models that can be generated according to the assumptions based on backward and forward windows. In the first assumption, we assume that the response time of B matches the time of its occurrence after A, and the time of its occurrence with respect to B. Therefore, C can be seen as a possible consequence of either A or B. In the same time the occurrence of B after A falls within its response time Γ_B, and therefore B can be possibly a consequence of A. As aforementioned, matching the timestamps of the change events do not necessarily mean they are correlated, and then we have to consider the possibility that the events might be independent. The possible propagation models are then depicted in the second column, which, merged together, give the probabilistic model in column 3. In the second assumption, we assume that C can not be candidate for A (according to its response time), and therefore the number of possible propagation models is reduced to only 4. In the last assumption, C can not be considered as consequence of both A and B, and then the number of models is reduced to 2.

To conclude, the forward and backward windows can be very useful in reducing the search space and highlighting the more probable propagation paths between the change events. In addition, the example of Figure 4 considers only a small window of events, where each event type occurred only once. In a bigger log, a same change event (e.g. a Replace on a partner) can occur several times, which may improve the precision of the probabilistic models.

Change Related Heuristics (CRH): The calculation of the prediction model could benefit from the relationships between change operation types. Indeed, from our experience [4], and considering solely the structural and public impacts, an initial change request of type Insert always generates insertions on the affected partners, and the same holds for Delete which generates only deletions. However, the Replace could generate all (three) types of changes. From this we deduce, that we can not have propagation of the type $(Insert, \pi_1) \rightarrow (Delete \vee Replace, \pi_2)$ or $(Delete, \pi_1) \rightarrow (Insert \vee Replace, \pi_2)$. Using these as punishments, the mining techniques could reduce the search space and therefore avoid incorrect behavior.

Choreography Model Heuristics (CMH): Another improvement consists in using the choreography model. The latter sketches all the interactions between the partners and gives a global overview on the collaboration structure. In this sense, we can use the dependencies between the partner interactions as a heuristic to identify transitive propagations (e.g. centrality). More details about heuristics that stem from the choreography structure can be found in [5]. These heuristics can be used to improve the mining results. For example, an identified **direct** propagation link between two partners through mining could be invalidated if the partners have no direct interactions together in the choreography model, and the change type is Delete. Because, unlike the Replace and Insert, the Delete does not result in new dependencies between the partners.



		partner1			partner2			partner3		
		INSERT	DELETE	REPLACE	INSERT	DELETE	REPLACE	INSERT	DELETE	REPLACE
partner1	INSERT	0	0	0	1	0	0	0	0	0
	DELETE	0	0	0	0	0	0	0	0	0
	REPLACE	0	0	0	1	1	0	1	0	0
partner2	INSERT	1	0	0	0	0	0	1	0	0
	DELETE	0	1	0	0	0	0	0	0	0
	REPLACE	1	0	0	0	0	0	0	1	0
partner3	INSERT	0	0	0	1	0	0	0	0	0
	DELETE	0	1	0	0	1	0	0	0	0
	REPLACE	0	0	1	0	0	0	0	0	0

(a)

		partner1			partner2			partner3		
		INSERT	DELETE	REPLACE	INSERT	DELETE	REPLACE	INSERT	DELETE	REPLACE
partner1	INSERT	0	0	0		0	0		0	0
	DELETE	0	0	0	0	?	0	0	?	0
	REPLACE	0	0	0		?	?		?	?
partner2	INSERT		0	0	0	0	0		0	0
	DELETE	0	?	0	0	0	0	0	?	0
	REPLACE		?	?	0	0	0		?	?
partner3	INSERT		0	0		0	0	0	0	0
	DELETE	0	?	0	0	?	0	0	0	0
	REPLACE		?	?		?	?	0	0	0

(c)

General Change Heuristics

0 (H1) if p1 == p2: then set all change types to 0

0 (H2) p1.INSERT: set p2.DELETE and p2.REPLACE to 0

0 (H3) p1.DELETE: set p2.INSERT and p2.REPLACE to 0

Structural Heuristics derived from Choreography Model

```
? (H4) p1.DELETE:
        if PARTNER_LINK(p1,p2) == 0 &&
          PARTNER_LINK(p2,p1) == 0 then
        set p2.DELETE to 0, otherwise maybe 1

? (H5) p1.REPLACE:
        if PARTNER_LINK(p1,p2) == 0 &&
          PARTNER_LINK(p2,p1) == 0 then
        set p2.DELETE to 0, otherwise maybe 1

? (H6) p1.REPLACE:
        if PARTNER_LINK(p1,p2) == 0 &&
          PARTNER_LINK(p2,p1) == 0 then
        set p2.REPLACE to 0, otherwise maybe 1
```

Fig. 5. (a) Genetic Encoding of a candidate solution (b) Candidate solution in graph form (c) Visualization of heuristics affecting a candidate solution

The implementation and the evaluation of the proposed heuristics within the memetic mining is described in the following sections.

5 Memetic Change Propagation Mining

In this section we outline the memetic algorithm for mining change event logs used to build change propagation models. This core algorithm is enriched with an implementation of the heuristics sketched out in the previous Section 4. Employing a change propagation model, predicting the behaviour of change requests in the choreography becomes possible. Memetic algorithms follow the basic flow of genetic algorithms (GAs) [12], which are stochastic optimization methods based on the principle of evolution via natural selection. They employ a population of individuals that undergo selection in the presence of variation-inducing operators, such as mutation and crossover. For evaluating individuals, a fitness function is used, which affects reproductive success. The procedure starts with an initial population and iteratively generates new offspring via selection, mutation and crossover operators. Memetic Algorithms are in their core GAs adding an inner local optimization loop with the goal of maintaining a pool of locally optimized candidate solutions in each generation [13].

Genetic Encoding: The genetic encoding is the most critical decision about how best to represent a candidate solution for change propagation, as it affects other parts of the genetic algorithm design. In this paper, we represent an individual as a matrix $\mathcal{D}$ that states the change relationships between the partners (the genotype). Each cell d_{ij} in the matrix has a boolean value equal to 1 only if a change on π_i is propagated to π_j, and zero otherwise. The matrix is non symmetric and the corresponding graph for change propagation is directed. This means that the probabilities of propagating changes from π_i to π_j and from π_j to π_i are not equal. This is due to the fact that the log file may contain more change propagations from π_i to π_j than from π_j to π_i. Figure 5(a) shows the representation of a candidate solution. Internally, the table rows are collapsed resulting in a *bitstring* of length $(m \times n)^2$ where n is the number of partners and

m is the number of change operation types (e.g., $(3 \times 3)^2$ in the Figure 5(a)). Figure 5(b) represents the corresponding propagation model graph. Figure 5(c) shows the importance of the heuristics in reducing the search space and their effects on candidate solutions.

Initial Population Generation: Two approaches are applicable for generating an initial population: (i) starting with random change propagation models by defining random paths for propagating change requests in each of the partners. This generated model may not respect the dependencies defined by the choreography model. Also considering the complexity of the problem caused by several constraints, the obtained results prove to be not sufficient. (ii) starting with an initial good solution using a part of the propagation dependencies existing in the log file. This solution could represent an incomplete behavior since some propagation paths of the log are not incorporated into the model. For the implementation we have chosen approach (ii) as it allows us to start with an approximate solution using that as the basis for search space exploration.

Heuristics: Here we briefly outline the implemented heuristics extracted from Section 4.

– **H1 (CMH)** - A change to a partner's process (π_1) never results in a propagation to him- or herself (e.g. $\pi_1 = \pi_2$). We can avoid solutions with this property in the search space by applying this heuristic.
– **H2 (CMH)** - Through our extensive simulations we can rule out the case where the originating change is of the type Insert and where the propagated change type is anything other than Insert (e.g. Delete and Replace).
– **H3 (CMH)** - Similarly to H2 we can rule out the cases where the originating change type is Delete, and the propagated change type is anything other than Delete (e.g. Insert and Replace).
– **H4 (CRH)** - Rule out propagated changes of type Delete $\Longleftrightarrow$ the originating change is of type Delete, and there is no interaction between the two partners (i.e. $\mathcal{R}(\pi_1, \pi_2) \cap \mathcal{R}(\pi_2, \pi_1) = \{\emptyset\}$, $\mathcal{R}$ is defined in Section 2).
– **H5 (CRH)** - Rule out propagated changes of type Delete $\Longleftrightarrow$ the originating change is of type Replace, and there is no interaction between the two partners (i.e. $\mathcal{R}(\pi_1, \pi_2) \cap \mathcal{R}(\pi_2, \pi_1) = \{\emptyset\}$).
– **H6 (CRH)** - Rule out propagated changes of type Replace $\Longleftrightarrow$ the originating change is of type Replace, and there is no interaction between the two partners (i.e. $\mathcal{R}(\pi_1, \pi_2) \cap \mathcal{R}(\pi_2, \pi_1) = \{\emptyset\}$).
– **H7** - The mutation as well as the crossover operation change a solution candidate in random ways. We can limit candidates of lower quality by taking into consideration only those events where both the partner and the change operation type occur (in pairs) in the change event log.
– **H8 (TRH/WRH)** - Both the timestamp and the window related heuristics are implemented as H8. The goal of both is to probabilistically find the correct (i) affected events given an originating event and (ii) originator given an affected event. This is accomplished via the forward (i.e. ω_i^+) as well as the backward window (i.e. ω_i^-) concept to limit the filtering process

for the most probable candidate events. Both windows are determined by $Max_{i \in \mathcal{P}}(\varGamma_i)$, i.e. the maximum average response times over all partner response times. For change event candidate selection inside the window, the individual timestamps are used to determine $\varDelta$. For the actual selection, the variance value ϵ can be determined empirically. We have opted to base this value on the candidate partner's average response time.

Fitness Function: The fitness function measures the quality of a solution in terms of change propagation according to the change event log CEL. The fitness score is a major component in (i) parent selection, determining each individual's eligibility for generating offspring and (ii) survival selection, determining which individuals survive into the next generation to pass on their genes. The following scoring logic is implemented as the fitness function as follows.

$$fitness = w_1 \times completeness + w_2 \times precision \tag{1}$$

Where w_1 and w_2 are weights, the completeness privileges individuals that are more complete according to the CEL and the precision penalizes propagation models that have extra behavior according to the CEL. We define ξ_{ij} as the probability that a change event type (i.e., Replace, Insert or Delete) on partner j is a consequence of a change event type on i. This probability is calculated based on the backward and forward windows, weighted by the number of occurrences of the same sequence of changes in the CEL. As illustrated in Figure 4, the individuals are probabilistic models that represent all or a subset of the change events of the CEL. A node in the propagation model σ is represented by a tuple (δ, π) containing a change type and a partner. The propagation probability between nodes i and j in σ is equal to ξ_{ij}. Then the completeness is given by the following weighted equation:

$$completeness = w_{11} \times \frac{\sum_{i \in \sigma}(\delta_i, \pi_i)}{\sum_{i \in CEL}(\delta_i, \pi_i)} + w_{12} \times \sum_{i,j=1..n} \xi_{ij} \times \phi_{ij} \quad with\ \phi_{ij} = \begin{cases} 1 & if (i,j) \in \sigma \\ 0 & otherwise \end{cases} \tag{2}$$

The first term concerns the percentage of traces of the CEL that are represented by the predicted model, while the second term calculates the percentage of the identified correlations between change events in the CEL that are considered by the propagation model. The attributes w_{11} and w_{12} are the weights given to each term of the equation.

$$precision = \sum_{i,j=1..n} (k \times \xi_{ij} - 1) \times \phi_{ij} \quad with\ \phi_{ij} = \begin{cases} 1 & if (i,j) \in \sigma \\ -1 & otherwise \end{cases} \tag{3}$$

The precision penalizes individuals with extra behavior (noise), by accumulating appropriate negative scores for propagation paths with very low propagation probabilities. The variable k is used to define when an event is considered to be noise; e.g., k=5, means propagation edges with probabilities less then 1/5=0.2 are considered as noise. Therefore, models containing several propagation paths with probabilities lower than 0.2 are classified as bad models. In this equation, we used a simple linear function to penalize noise $k \times \xi_{ij} - 1$, but can be changed to a more complex function (e.g. logarithmic). k is determined empirically.

6 Discussion and Evaluation

In this section we briefly describe the data set as well as the experimental setup in order to evaluate the memetic change mining algorithm coupled with the propsed heuristics for building change propagation models from change event logs (σ_{CEL}).

6.1 Data Set

The data used in this paper are obtained through our C^3Pro change propagation prototype[1]. During the simulation process, we generated change requests of type Replace, Insert and Delete. In total, 17068 change requests were created with an average of 682.7 requests per partner resulting in 49754 change propagation records in the CPL with an average of 2.9 derived propagations per initiated change request. In total 66822 change event records were generated and logged in the CEL. The logged data are in CSV format.

6.2 Benchmark Results

For the benchmark, the goal was on one hand to observe the effects of the applied heuristics on reducing the search space and on the other hand to validate the resulting mined model. Towards the former, each inclusion of a new heuristic should increase the maximal achievable score, as it takes less time to find an improved candidate solution. The benchmark was conducted in the following manner: (1) We start with creating distinct CEL slices with differing partner sizes of the range $[3, 16]$. $\Lambda = \{\lambda_i\}^{i \in [3,16]}$ (2) Then we define the heuristic sets to benchmark. The heuristic set *None* means we do not apply any heuristics, which practically reduces the memetic algorithm into a genetic algorithm. A heuristic set *H1-H3* means we apply heuristics *H1*, *H2* and *H3* within the local

Table 3. Benchmark Results: Memetic Mining of Change Logs. P=Partner Size. G=Generation. Score values are in range $[-\infty, 1]$ (inclusive), where 1 represents the best possible validation score.

	P=3			P=9			P=15		
	G=1	G=10	G=20	G=1	G=10	G=20	G=1	G=10	G=20
Heuristics									
None	-48.74	-48.26	-27.80	-268.75	-226.71	-186.64	-553.90	-502.61	-476.96
H1	-38.08	-30.03	-22.96	-241.34	-197.63	-168.61	-520.53	-482.92	-449.32
H1-H2	-25.94	-20.24	-17.17	-138.19	-106.18	-88.86	-337.45	-289.14	-250.25
H1-H3	**0.73**	**0.78**	**0.80**	0.50	0.55	0.56	0.54	0.55	0.55
H1-H4	0.72	0.76	0.75	0.53	0.58	0.55	0.57	0.55	0.61
H1-H5	0.72	0.77	0.75	0.56	0.50	0.64	0.60	0.62	0.56
H1-H6	0.72	0.77	**0.80**	0.50	0.63	0.63	0.62	0.65	0.67
H1-H7	0.72	**0.78**	0.77	**0.65**	0.67	0.56	**0.71**	**0.73**	0.72
H1-H8	0.71	0.77	0.75	**0.65**	**0.68**	**0.70**	**0.71**	**0.73**	**0.75**

[1] http://www.wst.univie.ac.at/communities/c3pro/index.php?t=downloads

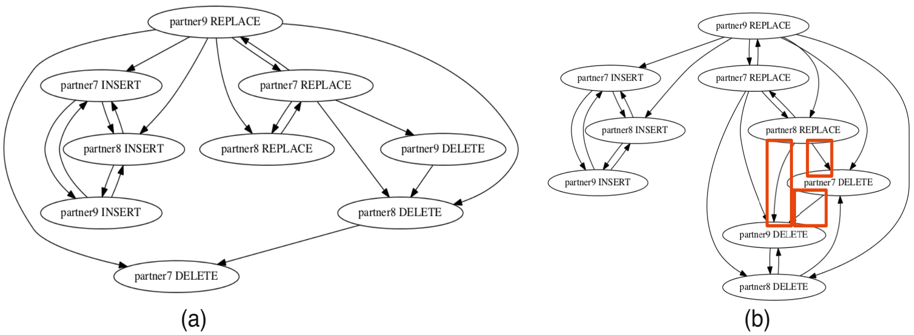


Fig. 6. Mined change propagation models (a) via CPL and (b) via CEL

optimization loop of the memetic algorithm. We have several such heuristic sets as can be observed by the rows in Table 3. (3) Each heuristic set is loaded into the memetic algorithm and executed on the change logs in Λ for up to 20 generations in turn. (4) For validating the mined model, we derive a propagation model from the CPL (i.e. σ_{CPL}), and compare it to the mined model (i.e. σ_{CEL}) by applying the following scoring function:

$$fitness_{validation} = completeness \times precision - penalites \tag{4}$$

$$precision = \frac{Nb\_extra\_propagation\_paths}{Nb\_total\_propagation\_paths} \tag{5}$$

$$completeness = \frac{Nb\_valid\_traces}{Nb\_total\_traces} \tag{6}$$

This validation function returns score values in the range $[-\infty, 1]$, where 1 represents the best possible validation score, meaning the two models are identical. We repeat this process ten times, storing the average score into the respective cells in Table 3. Underlined values are the best scores identified for each column.

We can generally observe the following: regardless of the employed heuristics, with each increasing partner size we obtain a lower quality candidate, except in the case where H8 is introduced. This behaviour signals the positive effects of time related heuristics (TRH) as well as window related heuristics (WRH). Similarly, with each increasing generation, the best candidate solution score increases. This holds true, except in cases where the survival selection routine (tournament selection) misses the current best candidate solution, resulting in a lower fitness score. Finally, we can indeed conclude that the proposed heuristics reduce the search space, as the quality of the best candidate solutions increase as more heuristics are added to the memetic change mining algorithm.

In terms of validation, Figure 6(b) shows the mined change propagation model using the described memetic change mining algorithm (on the CEL) with parameters: *partners* = 3, heuristics H1-H6 applied, and *generation* = 20. In contrast, Figure 6(a) represents the change propagation extracted from the CPL. According to Table 3, the average validation score of these two models are 0.72.

The differences between these two models are visually illustrated in the annotations in Figure 6(b). As can be seen in that figure, the memetic change mining algorithm could find a good approximation for the prediction model, showing only three extraneous edges (i.e. $(Replace, \pi_8) \rightarrow (Delete, \pi_9)$, $(Replace, \pi_8) \rightarrow (Delete, \pi_7)$ and $(Delete, \pi_7) \rightarrow (Delete, \pi_9)$). Our proposed memetic change mining algorithm fared well in this instance. As more partners are added, more candidates are included as potential consequences, leading to bigger models with more extraneous edges.

7 Related Work

Only few approaches have been proposed to compute the changes and their propagation in collaborative process settings [4,14,15,16]. Most of these approaches use either the public parts of the partner processes or the choreography/collaboration model; i.e., the global view on all interactions, to calculate the derived changes. They mainly calculate the public parts to be changed, but cannot anticipate the impacts on the private parts, which in turn, could engage knock-on effects on other partners. Besides, in some collaboration scenarios, a partner may have access to only a subset of the partner processes, and consequently could not estimate the transitive effects of the change propagation.

Change impact analysis has been an active research area in the context of large complex systems and software engineering [17,18,19,20]. As pointed out in [5], we studied these approaches, but found major differences to the problem discussed in this paper. One difference is based on the different structure of the underlying systems. Moreover, the use of the structured change propagation logs combined with memetic as well as genetic mining has not been employed before in these fields.

There exist approaches on impact analysis of change propagation within choreographies, i.e., [19,21]. However, they do not consider previous change propagation experience to enhance the prediction models.

Also they do not take into consideration the different metrics related to the specific structure of business process choreographies. Our previous work [5] on analyzing change impacts in collaborative process scenarios is based on the choreography structure only, i.e., it does not take into consideration any information on previously applied changes.

8 Conclusion

Being able to predict the change propagation behavior in collaborative process scenarios can contribute to time as well as cost reductions, which can determine the overall success of the cooperative process execution. Towards this end we have shown a memetic change mining approach for building a posteriori prediction models based on change event logs (CEL). This approach helps in cases where change propagation logs (CPL) (i.e. those logs which include complete propagation information) are lacking. In addition to the CEL as input, we have proposed a set of heuristics embedded in the memetic change algorithm to guide the candidate selection process towards higher quality ones. The conducted benchmarks and validation of the mined models (see Table 3) show the positive effects of the defined heuristics for reducing the search space, thus reducing the exploration time for finding accurate prediction models. Future work aims at mining change propagation logs (CPL), and analyzing dynamic impacts of process choreography changes.

References

1. Wynn, D.C., Caldwell, N.H.M., Clarkson, J.: Can change prediction help prioritize redesign work in future engineering systems? In: DESIGN, pp. 600–607 (2010)
2. Maier, A., Langer, S.: Engineering change management report 2011. Technical University of Denmark, DTU (2011)
3. Ahmad, N., Wynn, D., Clarkson, P.J.: Change impact on a product and its redesign process: a tool for knowledge capture and reuse. Research in Engineering Design 24(3), 219–244 (2013)
4. Fdhila, W., Rinderle-Ma, S., Reichert, M.: Change propagation in collaborative processes scenarios. In: IEEE CollaborateCom, pp. 452–461 (2012)
5. Fdhila, W., Rinderle-Ma, S.: Predicting change propagation impacts in collaborative business processes. In: SAC 2014 (2014)
6. Rinderle, S., Jurisch, M., Reichert, M.: On deriving net change information from change logs – The Deltalayer-Algorithm. In: BTW, pp. 364–381 (2007)
7. Günther, C., Rinderle-Ma, S., Reichert, M., van Der Aalst, W., Recker, J.: Using process mining to learn from process changes in evolutionary systems. International Journal of Business Process Integration and Management 3(1), 61–78 (2008)
8. Dustdar, S., Hoffmann, T., van der Aalst, W.M.P.: Mining of ad-hoc business processes with teamlog. Data Knowl. Eng. 55(2), 129–158 (2005)
9. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes, 1st edn. Springer (2011)
10. Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: Mining configurable process models from collections of event logs. In: Daniel, F., Wang, J., Weber, B. (eds.) BPM 2013. LNCS, vol. 8094, pp. 33–48. Springer, Heidelberg (2013)
11. Gaaloul, W., Gaaloul, K., Bhiri, S., Haller, A., Hauswirth, M.: Log-based transactional workflow mining. Distributed and Parallel Databases 25(3), 193–240 (2009)
12. Goldberg, D.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co. (1989)
13. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Natural Computing. Springer, Berlin (2007)
14. Rinderle, S., Wombacher, A., Reichert, M.: Evolution of process choreographies in DYCHOR. In: Meersman, R., Tari, Z. (eds.) OTM 2006. LNCS, vol. 4275, pp. 273–290. Springer, Heidelberg (2006)
15. Fdhila, W., Rinderle-Ma, S., Baouab, A., Perrin, O., Godart, C.: On evolving partitioned web service orchestrations. In: SOCA, pp. 1–6 (2012)
16. Wang, M., Cui, L.: An impact analysis model for distributed web service process. In: Computer Supported Cooperative Work in Design (CSCWD), pp. 351–355 (2010)
17. Bohner, S.A., Arnold, R.S.: Software change impact analysis. IEEE Computer Society (1996)

18. Giffin, M., de Weck, O., Bounova, G., Keller, R., Eckert, C., Clarkson, P.J.: Change propagation analysis in complex technical systems. Journal of Mechanical Design 131(8) (2009)
19. Oliva, G.A., de Maio Nogueira, G., Leite, L.F., Gerosa, M.A.: Choreography Dynamic Adaptation Prototype. Technical report, Universidade de São Paulo (2012)
20. Eckert, C.M., Keller, R., Earl, C., Clarkson, P.J.: Supporting change processes in design: Complexity, prediction and reliability. Reliability Engineering and System Safety 91(12), 1521–1534 (2006)
21. Wang, S., Capretz, M.: Dependency and entropy based impact analysis for service-oriented system evolution. In: Web Intelligence, pp. 412–417 (2011)