A STRATEGY FOR EVALUATING THE QUALITY OF TRACE ALIGNMENT TOOLS BASED ON A MARKOV MODEL

Xiao Bo
Rutgers University
Dep't of Electrical & Computer Eng.
Piscataway, NJ 08854
seanbo.cd@gmail.com

Ivan Marsic
Rutgers University
Dep't of Electrical & Computer Eng.
Piscataway, NJ 08854
marsic@rutgers.edu

Randall S. Burd
Children's National Medical Center
Division of Trauma & Burn Surgery
Washington, DC 20010
rburd@childrensnational.org

ABSTRACT
Trace alignment of event logs is used to understand and improve business processes. A key missing component of current approaches for performing trace alignment is a methodology to measure the quality of alignment. We propose a novel approach for generating random event logs that can be used for testing and evaluating trace alignment tools. Our results are relevant not for only validation of trace alignment tools but also for other process mining tools.

Categories and Subject Descriptors
Information Systems, Performance evaluation (efficiency and effectiveness)

General Terms
Algorithms, Measurement, Performance, Design, Verification.

Keywords
Trace alignment, algorithm validation, process mining

1. INTRODUCTION

1.1 Background and Motivation
In many processes, events can be recorded in an event log in which the events within a particular process execution ("trace") and their time of execution ("timestamp") are recorded. The event log can then be used to determine the pattern of sequences in which a process is executed. Determining these sequences is particularly challenging for complex processes for which the steps of the process are not well-known or when real-world uncertainties introduce a large amount of variability in process execution. Mining complex event logs for execution patterns and non-standard activities, however, may be important for understanding workflow and improving process quality and productivity.

Process mining approaches have been developed for identifying successful and unsuccessful patterns of process execution and identifying deviations within these processes that may be associated with adverse events [19]. A recent process mining approach for performing this type of analysis is "trace alignment," where the traces of events are aligned to identify the flow of events corresponding to the standard practice (known as "consensus sequence") [3]. Because patient-evaluation tasks during trauma resuscitation are standardized and prioritized, it is an appropriate approach to apply trace alignment on resuscitation event logs. Medical experts could use trace alignment techniques to visualize the event traces and discover the consensus sequence for idenfying the "average" case and deviations from this average. Trace alignment is widely used for studying genetic sequences and there are well-established techniques for evaluating the quality of trace alignment algorithms [22]. Trace alignment of business process logs, however, is a new technique and approaches for evaluating the quality of the output have not been developed. This paper focuses on developing a quality measure for trace alignment algorithms.

Our problem domain is trauma resuscitation—the initial care of severely injured patients in the emergency department. In trauma resuscitation, the ordering of tasks for evaluating the patient is essential to appropriate identification and management of potentially life-threatening injuries. It has been observed that critically-injured patients have up to a four-fold higher risk of death from errors than general hospital patients [17], with nearly half of these preventable deaths related to errors that occur during the initial resuscitation phase of treatment [7][12]. To standardize the resuscitation process and reduce variability, trauma centers around the world have adopted a protocol (the Advanced Trauma Life Support protocol [ATLS]), which prioritizes evaluation and management tasks [9]. The key components of the first phase of ATLS (the primary survey) can be followed using the acronym "ABCDE", representing the steps of airway evaluation (airway or "A"), evaluating respiratory mechanics (breathing or "B"), assessing and managing circulatory status (circulation of "C"), evaluating neurological status (disability of "D") and ensuring the patient is properly exposed (exposure or "E"). Given the potential benefits of ensuring ATLS protocol compliance, our long-term goal is to build an automatic decision-support system that will help reduce human errors during trauma resuscitation.

1.2 Problem Statement
Trace alignment strategies have been developed for business process mining, but these strategies can yield diverse results. No established methods exist for evaluating the quality of these alignment results. As an example, we performed an alignment of 268 ATLS traces for a set of ATLS tasks using two different trace alignment strategies using a plugin in ProM 6.3[1] (Figure 1). Despite using the same initial data set, different results were obtained when a block-shift alignment algorithm (right of Figure 1) or regular alignment algorithm [3] that does not use block-shift alignment (left of Figure 1) was used. As an example, one trace

[1] ProM 6.3 website: http://www.promtools.org/prom6/

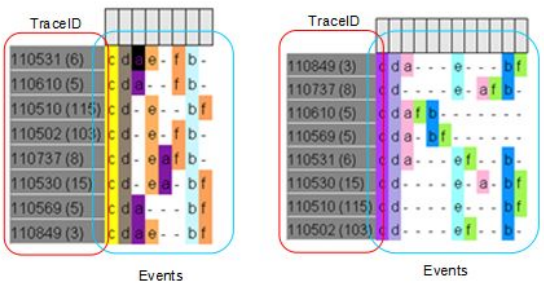

Figure 1: Example trace alignments for the same event log using two different algorithms available in ProM 6.3. The left chart was produced using a regular alignment algorithm and the right chart additionally used the block shift algorithm. The letters stand for: c = airway assessment, d = breath sounds, a = palpation of central pulses, e = palpation of distal pulses, b = Glasgow Coma Score (GCS) assessment, f = pupil exam.

("110531" in Figure 1) had a different length and position of gaps (" – ") depending on the algorithm used. Because of how differences in length and position of gaps that occur with each algorithm, two alignments and their consensus sequences are different.

Given that the results of trace alignment may differ based on the method used, it is necessary to have a benchmark that can measure the quality of output of each trace alignment algorithm. We propose that a tool that applies this type of a benchmark should address the following two issues:

- *Generating Artificial Event Logs*: The tool should be able to generate random event logs based on the process model. During a business process execution, the process may not follow the standard ordering of tasks because of errors or other reasons. The tool should be able to reproduce the kind of variability present in real executions of the process.
- *Consensus Sequence Evaluation*: The tool should be able to evaluate the quality of the computed consensus sequence by comparing it with each trace in the input event log.

1.3 Related Work

In the area of process mining, the main approach to creating benchmark datasets is based on a generative framework for representing well-understood processes (e.g., PLG Framework [5]). The approach is based on a context-free grammar of commonly observed patterns of event sequences, such as "sequence," "split and merge," or "loop." Each production of this grammar is associated with a probability of occurrence of a pattern in the current step. This approach may be suitable for describing well-understood existing processes or newly designed processes. It may not, however, be suitable for complex and poorly understood processes such as trauma resuscitation. Because process patterns and corresponding probability distributions are unknown, the generated event log may not be a realistic representation of actual process executions.

ROSE [18] is an approach that was developed for generating test datasets of DNA, RNA, and protein sequence alignments. This approach is based on a phylogenetic evolution tree and a substitution model derived from known statistics of evolution processes at the molecular sequence level. It also considered insert-delete (INDEL) operations and "sequence motifs" which represent regions of functional importance, in which mutation is less likely to happen. The knowledge of these facts is critical to

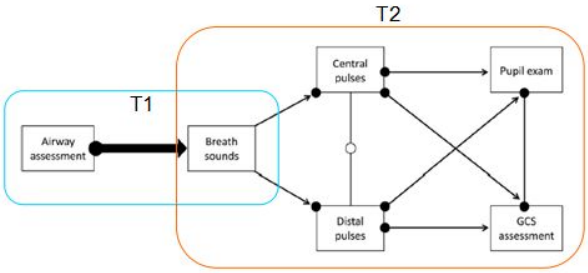

Figure 2: Declarative model of six key tasks of the ATLS primary survey.

simulating the evolution process. A limitation of this approach is that it mainly focuses on the evolution of each position in a given sequence but overlooks the dependencies between adjacent positions. Each position is mutated independently of the adjacent positions, reflecting the nature of genetic mutations. In contrast to this approach, the next task in a business process often depends on the current task. For this reason, we need to consider the task in the current position when selecting the "mutation" for the next position (the next task).

Related to the second issue of consensus sequence evaluation, two main groups of string or sequence metrics can be used [16]. The first group of metrics includes the Hamming distance, which is computationally very fast and has a low memory requirement but may not reflect the actual similarity between two sequences. The second group includes the edit distance (based on Levenshtein distance), which can better measure the similarity between two sequences but is less efficient in terms of processing speed and memory requirements.

This paper addresses the two key issues identified earlier. First, we propose a novel method for generating artificial event logs of the trauma resuscitation process that can serve as test event logs for trace alignment tools. Our method is based on an equivalent of phylogenetic tree tree in genetics. We can think of the "ideal" execution of the resuscitation process that is prescribed by ATLS as equivalent of the "ancestor sequence." Any variations observed in actual resuscitations can be thought of as "mutations of the ancestor sequence." The statistical parameters of our method are derived from real resuscitations logs. Second, we propose a methodology to measure the quality of consensus sequences based on Phred-quality score [10] and edit distance.

2. PROCESS MODEL AND DATA SOURCE

2.1 Data Source

Our input event log is generated from video recordings at the trauma center of the Children's National Medical Center (CNMC) in Washington, DC. The data were collected over from events that occurred over eight months. The resulting log contains ATLS event traces from 437 trauma resuscitations as well as a timestamp allowing assessment of the order of event performance [8]. The approach that was used for obtaining this event log has been previously described [13][14].

2.2 Declarative Model and Statistical Model

A declarative model of the process for which the traces were collected was developed by medical experts based on the knowledge of the ATLS process [2] performed by the bedside evaluator (Figure 2). It included six primary ATLS survey tasks. This model represents the best practice for ordering and inclusion of these tasks. For example, "breath sounds" should not happen

Table 1: Contents of T1, where the letter codes are as defined in Figure 1. T1[c,d] means the probability of position 1 being "d=breath sounds" if position 0 was "c=airway assessment."

PREVIOUS \ NEXT	c	d	a	e	b	f
c	0	0.95	0.01	0.02	0.02	0
d	0.6	0	0.03	0.3	0.03	0.03
a	0	0	0	0	0	0
e	0	0	0	0	0	0
b	0.7	0.3	0	0	0	0
f	1	0	0	0	0	0

before "airway assessment" is accomplished and "GCS assessment" should not occur unless either the "central pulse" or "distal pulses" are measured. Although this model represents best-practice, actual process execution may not follow this model. First, the provider may go back to previous steps to reevaluate an observed change. Second, the process may be performed out of order because of situational variables such as the availability of needed evaluation tools or personnel.

We can state the above description in a more formal way by representing a process using a first-order Markov model. Given a trace with M events, the i^{th} event ($1 \le i \le M$) depends on the $(i-1)^{st}$ event, for $i>1$. The parameters of the proposed model are as follows:

- *Event frequencies* PREFIX_FREQ(f_1, f_2, ..., f_n) satisfying $\sum_{i=1}^{n} f_i = 1$ are used for creating the first event in a trace.
- *Mutation matrix* T1 represents the transition probability from the first to the second event.
- *Mutation matrix* T2 represents the transition probability from event i to event $i+1$ ($i \ge 2$).
- *Length frequencies* L(ℓ_{E1}, ℓ_{E2} , ..., ℓ_{En}), represent the possible lengths of traces that start with different events $E1$, $E2$, ..., En. Given a first event of type Ei in a trace, different possible lengths k of the trace occur with different frequencies f_k. Therefore, $\ell_{Ei}(f_1, f_2, ..., f_n)$ represents the probability distribution of different trace lengths, given that the first event = Ei and the distribution satisfies $\sum_{k=1}^{n} f_k = 1$. Let us assume that an event trace could start with these first events: $E1$="airway assessment" or $E2$="breath sounds." We then need to specify L = $\{\ell_{airway\text{-}assessment}, \ell_{breath\text{-}bounds}\}$. For example, traces that start with "airway assessment" and may be up to five events long, but only trace lengths 2 and 5 occur, and they occur equally likely. The probability distribution for $\ell_{airway\text{-}assessment}$ = $\{f_1 = 0, f_2 = 0.5, f_3 = 0, f_4 = 0, f_5 = 0.5\}$.

The motivation to partition the mutation matrix into T1 and T2 is to avoid the possibility that noise present in T2 affects T1. For example, based on the declarative model (Figure 2), "pupil exam" and "GCS assessment" are not acceptable in the first two steps, but are highly coupled. Based on the event log of 437 observed traces, their mutual transition probabilities after the first two steps ("pupil exam → GCS assessment" and "GCS assessment → pupil exam") are ≥0.85. However, some real traces started with "pupil exam" (not adhering to the model in Figure 2), which was invariably followed by "airway assessment" (see that T1[f, c] = 1 in Table 1). In these types of cases, if we combined all transition probabilities into a single mutation matrix, the system could generate a trace similar to "pupil exam → GCS assessment → pupil exam → GCS assessment", which is unacceptable because no such traces were observed. In other words, statistics from the "normal" cases (represented by the model) could distort the transition probabilities.

The model parameter PREFIX_FREQ(f_1, f_2, ..., f_n) is extracted from a given event log as follows:

Table 2: Contents of T2, where the letter codes are as defined in Figure 1. T2[c,d] means the probability of i^{th} event = "d=breath sounds" if $(i-1)^{st}$ event "c=airway assessment," where 1<i<trace-length.

PREVIOUS \ NEXT	c	d	a	e	b	f
c	0	0.01	0.06	0.52	0.16	0.16
d	0.003	0	0.09	0.79	0.07	0.05
a	0.02	0.04	0	0.18	0.41	0.43
e	0.04	0.002	0.1	0	0.41	0.43
b	0.004	0.03	0.05	0.07	0	0.85
f	0.02	0	0.07	0.06	0.85	0

$f_i = \frac{t_1}{t_2}$, where t_1 is the number of traces starting with event type i and t_2 is total trace number.

The followings are corresponding pseudo code:

```
Function get_prefix_freq(log) return PREFIX_FREQ:
PREFIX_FREQ = {}    // PREFIX_FREQ is a Python-like
        // dictionary. It could also be implemented by a hash-table.
Counter = 0
For trace in log:
   First_event = trace[0]
   If first_event not in PREFIX_PREQ:
      PREFIX_FREQ[First_event] = 0
   End if
   PREFIX_FREQ[First_event]++
   Counter++
End for
For event in PREFIX_FREQ:
      PREFIX_FREQ[event] = PREFIX_FREQ[event] / Counter
End for
Return PREFIX_FREQ
```

Other parameters (e.g. T1, T2, L) of the model are extracted similarly. Table 1 and Table 2 show the extracted contents of T1 and T2. The letter encoding is as defined in Figure 1.

3. ALGORITHM

3.1 Overview

Our system consists of two components:

1. Trace Generator: creates a random event log used in trace alignment
2. Consensus Evaluator: evaluates the generated consensus sequence.

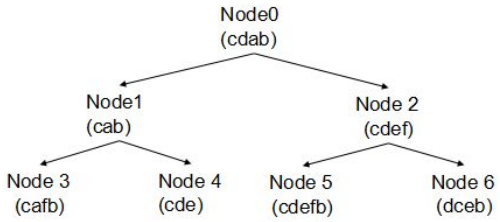

Figure 3: An example guide tree used for simulating real-world variability of trauma resuscitation. The letter encoding for example traces in the parentheses is as defined in Figure 1.

3.2 Trace Generator

The trace generator produces a random event log simulating real-world variability of the business process. It is common to represent this process by a guide tree, in which each node contains a trace (Figure 3). The root node (Node 0) contains an "ideal" trace of standard ordering of tasks, and the other nodes contain variations. When executing the process in a standard way, errors may occur in the actual execution and would "mutate" from Node 0 to Node 1. For instance, suppose the trace in Node 0 is "airway assessment → breath sounds → central pulse → GCS assessment." "Breath sounds" may have been skipped in the trace in Node 1 and become "airway assessment → central pulse → GCS assessment." As more mistakes occur, the actual execution may further "mutate" from Node 1 to Node 4.

In addition to (PREFIX_FREQ, T1, T2, L), the generator requires the following input:

- An Ancestor Trace *S*: This trace is the standard trace used to generate variations. This trace should be user-defined. If the user does not provide the standard trace (e.g., because of a poorly specified process), a random trace of length ℓ will be generated based on the model parameters (PREFIX_FREQ, T1, T2 and L).
- Edge Distance *D*: This number represents the frequencies of mutation from parent node to its children nodes. In Figure 3, if *D* = 8, Node 1 mutates eight times before reaching Node 4. The edge distance is used to create a mutation guide tree which acts as a mapping for generating a random event log [18].
- Revival Probability *R*: This number represents the probability to generate a child trace without considering the parent trace. For instance, Node 4 may be created as a random trace rather than a mutation of Node 1. The main motivation for this feature is to increase the randomness of output datasets.
- Trace Number *N*: This number determines the number of traces in the output event log E (defined below).
- Tree Depth *d*: This number represents the depth of the guide tree, i.e., how many generations the tree will evolve away from the root trace. For example, in Figure 3, *d* = 2.

The output of the trace generator is:

- Event Log *E*: The event log contains all random traces that are generated by the trace generator.

The following is the pseudocode of the algorithm:

Function trace_generator(S, D, d, N, R, PREFIX_FREQ, T1, T2, L) **return** E
If S is undefined
    S = create_sub_trace(null, 0, n, PREFIX_FREQ, T1, T2, L)
End if

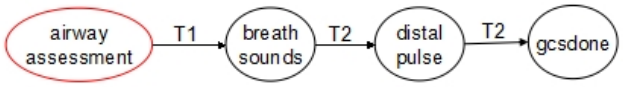

Figure 4: Generation of a sample sub trace. In this case, seed event *E* = airway assessment, start position *P* = 0, and trace length ℓ = 4.

T = create_guide_tree(D, d)
prune_tree(T, N)

T.root = S
fill_tree(T.root)
Traces = get_traces(T)
Return Traces

Function fill_tree() is implemented as a recursion:

Function fill_tree(T.root):
Search_motifs(T.root)
Fill_Nodes(T.root)
Return

Function Fill_Nodes(tree_node):

For Children of tree_node:

    Children.seq = mutate(tree_node)

    Search_motifs(Children)

    Fill_Nodes(Children)

End For

Return

The remaining functions are defined in the following subsections.

3.2.1 *Generating the Sub Traces*

The function create_sub_trace(E, P, ℓ, PREFIX_FREQ, T1, T2, L) is the core function for generating event logs. It returns a trace of length ℓ. In addition to (PREFIX_FREQ, T1, T2, L), it also requires the following inputs:

- Seed Event *E*: The first event in the trace or sub-trace that will be generated.
- Start Position *P*: The position of the seed event in the original trace. This number will determines what the transition model (T1, T2) will be used for generating a sub-trace.
- Trace Length ℓ: The length of the target (output) sub-trace.

Figure 4 shows an example creation of a sub-trace. Because the seed event is also the first event in the parent-node trace, the second event in a sub-trace is determined by E and T1 based on the model (Figure 2). The third event is based on T2 and the second event, and so is the fourth event. If the input seed event is null, the seed event will be generated from PREFIX_FREQ.

3.2.2 *Generating the Mutation Guide Tree*

Creation of the mutation guide tree is done in the function create_guide_tree(). After the tree is created, *N* nodes are randomly marked as "selected," because the user requested *N* traces in the output event log *E*. To reduce the computational complexity, we compute the minimum spanning tree and prune the tree. For example, in Figure 3, when Nodes 3 and 4 are

selected, Nodes 2, 5, and 6 will be removed because they are not relevant to reaching Nodes 3 and 4 from the root node. After the minimum spanning tree is generated, the system generates the output event log *E* from the root node by fill_tree() and get_traces().

3.2.3 *Generating Children Traces*

The core function for children trace creation is the function mutate(). The followings are its main steps:

1. Generate trace of length ℓ based on the type Ei of the first event in the trace: ℓ = getLength(Ei, L)
2. Determine the mutation start position STARTPOS (0≤STARTPOS<ℓ)
3. Create sub-trace based on T_{old}[STARTPOS]:

   T_{sub} = create_sub_trace(T_{old}[STARTPOS], STARTPOS, ℓ – STARTPOS, PREFIX_FREQ, T1, T2, L)
4. Create child trace by removing the old sub-trace starting at position STARTPOS and instead appending the created sub-trace to it:

   T_{old}.RemoveAll(STARTPOS, T_{old}.length);
   T_{new} = T_{old}.append(T_{sub})

By the above steps, the INDEL operations are realized because L contains all possible trace lengths. The sequence motif information is also represented in T1, T2. These steps are repeated for edge distance *D* times in order to maintain the randomness of child traces.

3.3 Consensus Sequence Evaluator

The quality of a consensus sequence is evaluated from two aspects: confidence about the element in each position and similarity between traces in event log and consensus sequence. The Phred-quality score is used to measure the confidence for each position:

$Q = -10 \cdot \log_{10}(P)$, where Q represents Phread-quality score and P represents *base-calling error probability* [10][11].

Q, however, only estimates the confidence of the element in each position in the consensus sequence, independently of other positions. For this reason, we use regular edit distance to measure similarity between the consensus sequence and individual traces in test event logs instead of Q. "Sequence motifs" which represent critical positions in the root trace that are subject to lower mutation rates compared to other positions, are only implicitly contained in our statistical model. If sequence motifs were to be explicitly considered as event patterns, we would need to add a "penalty" in the quality score formula, to penalize mutations in the corresponding positions. Because the ATLS model in Figure 2 is highly simplified and includes only six tasks that are equally important, we are deferring introducing "sequence motifs" for our future work.

4. RESULTS AND DISCUSSION

To test our trace generator, we used the concept of *Replay Fitness Score* [4][19]. This score is used to quantify the extent to which a model can reproduce the traces recorded in an event log. The score is computed as [4]:

$$Q_{rf} = 1 - \frac{\text{cost for aligning model and event log}}{\text{Minimal cost to align arbitrary event log on model and vice versa}}$$

where Q_{rf} is between 0 and 1. The value 1 means that the model can perfectly replay the event log and 0 means that the model cannot replay the log.

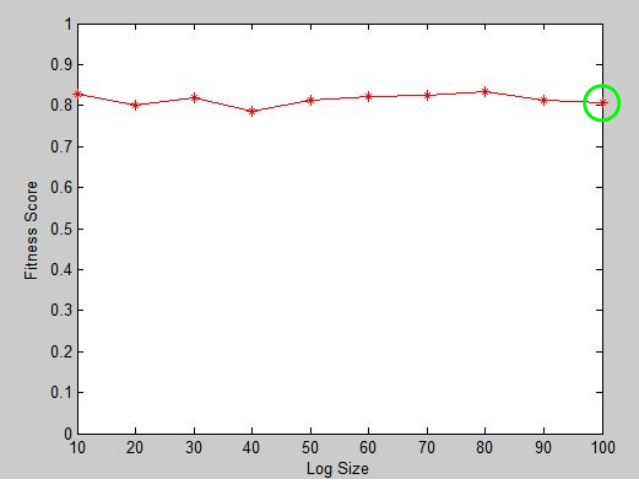

Figure 5: Replay Fitness Score of artificial event logs generated by our tool, compared to the Petri-net model created by ProM from 437 actual event traces.

To explore the properties of our trace generator, we generated 10 event logs with 10 to 100 traces each. The ancestor trace *S* is "airway assessment → breath sounds → pupil exam → GCS assessment → pupil exam → GCS assessment" and the guide tree is created with the parameters d = 10, D = 1 and R = 0.5 (Section 3.2). We also computed an approximate model (Petri Net) of 437 traces by using the ProM plugin "Mine a Petri net using ILP" [21]. We next computed the fitness score using the ProM plugin named "Replay a log on Petri Net for conformance checking." The fitness score determines how well our artificially generated event log fits the Petri Net model of the actual event log [1][20]. The core algorithm is a cost-based A* algorithm [1]. The fitness score Q_{rf} for our event log generator is around 0.8 (Figure 5). Based on the definition of Q_{rf}, it should be equal to 1 for all artificial event logs that perfectly simulate the real-world process execution. A potential reason that event logs generated by our tool generally have lower fitness score (Figure 5) is that the Integer Linear Programming (ILP) approach performs well in terms of speed but lacks in capability to handle complex and less structured process [21]; therefore, the Petri-net may not correctly reflect the process execution model.

To further understand the characteristics of our trace generator, we used one of event logs from Figure 5. This event log had 100 artificial traces. Its fitness score is circled by green in Figure 5. We computed its consensus sequence C1 using the trace alignment plugin of ProM as C1 = "airway assessment → breath sounds → GCS assessment → pupil exam." The plugin uses the Euclidean distance [8] as part of the alignment process. We also computed the consensus sequence C2 of the 437 actual traces using same algorithm and obtained C2 = "airway assessment → breath sounds → distal pulse → GCS assessment → Pupil Exam." When these two consensus sequences are compared, the only difference is that in C2, "distal pulse" is missing (Figure 6). Based on T2 (Table 2), both "GCS assessment" and "distal pulse" can follow after a "breath sounds" event, but the probability T2[breath sounds, distal pulse] is much greater than T2[breath sounds, GCS assessment]. Therefore, the consensus sequence C1 of our artificial event log should not have omitted "distal pulse" if our tool perfectly simulated the real-world process execution. Two potential explanations may account for this shortcoming. First, the trace alignment algorithm in ProM [3] may not be robust enough to handle some real-world

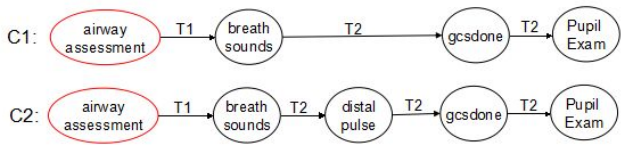

Figure 6: Comparison of the consensus sequence for an artificially-generated event log (C1) to that of the 437 actual event traces of trauma resuscitation (C2).

variability of event traces. Second, 100 traces in our artificial event log or 437 actual traces may be insufficient for accurate representation of statistical properties of the trauma resuscitation process.

We used Phred-quality score for consensus sequence evaluation, which has proved to provide a robust metric for evaluating the performance of alignment tools for genetic sequences [10][11]. Phred-quality score to measure the confidence about each position in a consensus sequence. Further work in this area will be a subject of our future work.

5. CONCLUSION AND FUTURE WORK

The event logs created by our methodology are artificial datasets used for evaluating how the ProM trace alignment tool performs on a given event log. Our experimental results showed that our tool can generate random event logs that can replay the real execution of a declarative model of ATLS. Our tool combines the strengths of several existing evaluation frameworks (e.g., PLG and ROSE), such as using a guide tree and process model to generate realistic event logs of the trauma resuscitation process. Unlike PLG, we use the statistical model extracted from real data and a guide tree to event traces. Unlike Rose, which mutates each position in the sequence independently of its neighbors, we consider the current task when generating the next task. In this sense, our tool "grows" the event traces from starting event to the final event. For these reasons, our tool is suitable for evaluating trace alignment tools for their potential use for studying event logs of trauma resuscitations.

Our model was designed to simulate the event traces resulting from real execution of an ATLS-based process. We plan to improve this evaluation tool in several ways. First, the model for the INDEL operation is unrealistically simple. Currently, we randomly choose a position in the trace and then grow a new sub-trace from it. However, not all positions have the same probability that a mutation will occur. For instance, about 78% of our dataset of 437 traces started with "airway assessment", and about 95% of those traces which started with "airway assessment" had "breath sounds" in the second position. Second, as discussed in Section 4, we need to show whether Phred-Quality Score is accurate for consensus sequence evaluation. Third, some combinations of tasks are critical and are less likely to vary (i.e., mutate), because the workers are trained to pay special attention to these tasks. To address this potential finding, we also plan to introduce domain-knowledge and define "motifs," and suppress mutation on these tasks.

6. REFERENCES

[1] Adriansyah, A., van Dongen, B.F., and van der Aalst, W.M.P., "Conformance checking using cost-based fitness analysis," ACSD 2011: 57-66.

[2] American College of Surgeons Committee on Trauma and Institutional Practice, ATLS: Advanced trauma life support for doctors (student course manual), 9th ed., American College of Surgeons, Chicago, IL, 2012.

[3] Bose, R.P.J.C., van der Aalst, W.M.P., "Process diagnostics using trace alignment: opportunities, issues, and challenges," Inform Syst 2012;37:117-141.

[4] Buijs, J.C.A.M., van Dongen, B.F., and van der Aalst, W.M.P., "On the role of fitness, precision, generalization and simplicity in process discovery," On the Move to Meaningful Internet Systems: OTM 2012.

[5] Burattin, A., and Sperduti, A., "PLG: A framework for the generation of business process models and their execution logs," BPM 2010 Workshops, LNBIP 66, pp. 214-219, 2011.

[6] Carter, E.A., Waterhouse, L.J., Kovler, M.L., et al. "Adherence to ATLS primary and secondary surveys during pediatric trauma resuscitation," Resuscitation 2013;84:66-71.

[7] Demetriades D, Kimbrell B, Salim A, Velmahos G, Rhee P, Preston C, Gruzinski G, and Chan L., "Trauma deaths in a mature urban trauma system: Is 'trimodal' distribution a valid concept?" J Am Coll Surg. 2005 Sep;201(3):343-348. PubMed PMID: 16125066.

[8] Deza, E., and Deza, M.M., Encyclopedia of Distances, p. 94, Springer-Verlag, 2009.

[9] Driscoll, P. and Wardrope, J: "ATLS: past, present, and future," Emerg Med J 2005;22:2-3 doi:10.1136/emj.2004.021212

[10] Ewing B., Green, P. "Base-calling of automated sequencer traces using Phred. II. Error probabilities," Genome Res. 8 (3): 186-194, 1998.

[11] Ewing, B., Hillier, L., Wendl, M.C., and Green, P., "Base-calling of automated sequencer traces using Phred. I. Accuracy assessment". Genome Res. 8 (3): 175-185, 1998.

[12] Gruen, R.L., Jurkovich, G.J., McIntyre, L.K., Foy, H.M., and Maier, R.V., "Patterns of errors contributing to trauma mortality: lessons learned from 2,594 deaths," Ann Surg. 2006 Sep;244(3):371-80. PubMed PMID: 16926563; PubMed Central PMCID: PMC1856538.

[13] Kelleher, D.C., Bose, R.P.J.C., Waterhouse, L.J., Carter, E.A., and Burd, R.S., "Effect of a checklist on advanced trauma life support workflow deviations during trauma resuscitations without pre-arrival notification," Journal of the American College of Surgeons, 2014 Mar;218(3):459-66. doi: 10.1016/j.jamcollsurg.2013.11.021. Epub 2013 Nov 26.

[14] Kelleher, D.C., Carter, E.A., Waterhouse, L.J., and Burd, R.S., "Compliance with barrier precautions during pediatric trauma resuscitations," Resuscitation, vol. 84, pp.314-318, 2013.

[15] Kelleher, D.C., Kovler, M.L., Waterhouse, L.J., et al. "Factors affecting team size and task performance in pediatric trauma resuscitation," Pediatr Emerg Care, In press.

[16] Seker, S.E., Altun, O., Ayan, U., and Mert, C., "A novel string distance function based on most frequent k characters", International Journal of Machine Learning and Computation (IJMLC), vol. 4, no. 2, pp.177-183, 2014.

[17] Stahl, K.D., and Brien, S.E., "Reducing patient errors in trauma care," In: Cohn SM, ed. Acute Care Surgery and Trauma Care. London, UK: Informa Health Care, 268-277, 2009.

[18] Stoye, J., Evers, D., and Meyer, F., "Rose: generating sequence families," Bioinformatics, vol. 14, no. 2, pp. 157-163, 1998.

[19] van der Aalst, W.M.P. Process Mining: Discovery, Conformance and Enhancement of Business Processes, Springer-Verlag Berlin Heidelberg, 2011.

[20] van der Aalst, W.M. P., Adriansyah, A., van Dongen, B.F., "Replaying history on process models for conformance checking and performance analysis," Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery 2(2): 182-192, 2012.

[21] van der Werf, J.M.E.M., van Dongen, B.F., van Hee, K.M., Hurkens, C.A.J., and Serebrenik, A. "Process discovery using integer linear programming," Applications and Theory of Petri Nets, vol. 5062, pp 368-387, 2008.

[22] Waterman, M.S., Introduction to Computational Biology: Maps, Sequences and Genomes, Chapman & Hall, CRC, 2000.