# A Multi-objective Approach to Business Process Repair

Chiara Di Francescomarino, Roberto Tiella, Chiara Ghidini, and Paolo Tonella

FBK-irst, Via Sommarive 18 Povo, 38050,Trento, Italy
{dfmchiara,tiella,ghidini,tonella}@fbk.eu

**Abstract.** Business process model repair aims at updating an existing model so as to accept deviant (e.g., new) behaviours, while remaining as close as possible to the initial model. In this paper, we present a multi-objective approach to process model repair, which maximizes the behaviours accepted by the repaired model while minimizing the cost associated with the repair operations. Given the repair operations for full process repair, we formulate the associated multi-objective problem in terms of a set of pseudo-Boolean constraints. In order to evaluate our approach, we have applied it to a case study from the Public Administration domain. Results indicate that it provides business analysts with a selection of good and tunable alternative solutions.

## 1 Introduction

*Business process model repair* can be used to automatically make an existing process model consistent with a set of new behaviours, so that the resulting repaired model is able to describe them, while being as close as possible to the initial model [4]. Differently from process discovery, in which a completely new process is discovered from the new observed behaviours, process model repair starts from an initial process model and it incrementally evolves the available model through a sequence of repair operations [4]. *Repair operations* range from simple insertion and deletion of activities in the model, to sophisticated sets of operations. In all cases, however, repair operations have a cost: they add complexity to the repaired models. Business analysts in charge of repairing existing models with respect to new behaviours are hence forced to choose whether to accept the increased complexity of a model consistent with all deviant behaviours, or to sacrifice consistency for a simpler model. In fact, some deviant behaviours may correspond to exceptional or error scenarios, that can be safely abstracted away in the process model.

In this work, we propose a multi-objective optimization approach to support business analysts repairing existing process models. It uses repair operations from state-of-the-art process repair algorithms to define a multi-objective optimization problem, whose two objectives are: (1) minimizing the cost of repair (in terms of complexity added to the repaired model); and, (2) maximizing the amount of new behaviours represented consistently in the model. We formulate such multi-objective optimization problem in terms of a set of pseudo-Boolean constraints and we solve it by means of a Satisfiability Modulo Theory (SMT) solver. The result provides business analysts with a set of Pareto-optimal alternative solutions. Analysts can choose among them based on the complexity-consistency trade-off that better fits their needs. The approach has been evaluated on a real life case study.

The contribution of the paper is twofold: (i) a multi-objective approach for business process model repair (Section 3); (ii) the results of our evaluation of the approach on a real-life case study (Section 4).

## 2 Background

Inputs to the automated process repair techniques are new process behaviours, which in modern information systems are captured through new execution traces recorded in log files, so the problem of automated repair can be stated as the problem of *repairing a process model with respect to a log file* [4]. In other words, given an *initial process model* $M$ (either manually designed or automatically discovered) and a set of execution traces $T$ (describing the new behaviours of the system), *automated process repair* aims at transforming $M$ into a new model $M'$ that is as close as possible to $M$ and that accepts all traces in $T$, where an execution trace $t \in T$ is a sequence of events (i.e., system activities) $t = \langle e_1, ..., e_n \rangle$.

Among the different ways in which automated repair can be realized, two main categories of approaches can be identified in the literature: (i) the approaches performing *repair operations* on the initial model $M$ by directly looking at its *differences* with the new traces [4]; (ii) the approaches that mine from $T$ one ($M_T$) or more ($M_T = \bigcup(M_{t_i})$) new process models describing the new behaviours, use delta-analysis [1] techniques for identifying differences between the new mined models and the initial one, $M$, and apply repair operations to $M$ [7,6].

In both cases, the *differences* of the initial process model with respect to the deviant behaviours (described as execution traces or as mined process models) have to be identified (see e.g., [4] and [7]). Once such differences have been identified, a set of *repair operations* can be applied to the initial model $M$. The basic operations consist of *insertion* and *deletion* of activities in the model. For example, given the extract of Petri Net in Figure 1 and the execution traces $t_1 = \langle A, B, D, C \rangle$ and $t_2 = \langle A, C \rangle$, two basic repair operations, an insertion $o_1$ and a deletion $o_2$ (see Figure 2) can be applied to the Petri Net in order to make $t_1$ and $t_2$ accepted by the Petri Net. Since these operations might remove old behaviours of the net, some approaches (e.g., [4]) tend to be conservative and to introduce the addition or removal of behaviours only as an optional alternative to the old behaviours. Figure 3 shows how this can be realized in a Petri Net: the black transitions represent silent transitions, i.e., transitions that are not observed when the net is replayed. Note that while preserving old behaviours, repair operations can introduce extra-behaviours such as the one described by the execution $\langle A, D, C \rangle$.

In this work we use a repair technique belonging to the first group of approaches (repairs based on trace differences) and, in detail, the ProM[1] *Repair Model* plugin. This plugin implements the approach proposed by Fahland et al. [4] and takes as input a Petri Net describing the initial model $M$ and a log. A cost is assigned to insertion and deletion operations. Correspondingly, an optimization problem is defined and the lowest-cost alignment between the process model and the set of input traces is computed. The outcome is a Petri Net $M'$ which is able to accept all traces in $T$.

[1] `http://www.promtools.org/prom6/`

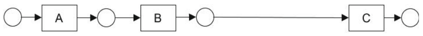

**Fig. 1.** An extract of $M$ described as a Petri Net

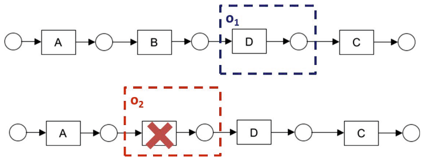

**Fig. 2.** Base repair operations

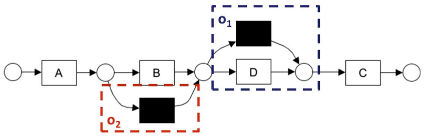

**Fig. 3.** Base operations preserving old behaviours

On top of this base alignment algorithm and of the insertion and deletion operations described above, a set of variations are proposed in the approach by Fahland [4]:

***Subprocess repair operations.*** In order to improve the precision of the repaired model $M'$, i.e., to avoid having too many extra-behaviours (besides those in $T$), a *subprocess repair operation* is introduced. The idea is that whenever a sequence of inserted activities occurs at the same place in the model, instead of adding these activities incrementally, they are structured as a subprocess, which is mined starting from the set of subtraces that maximize the sequence of skipped activities in $M$. For example, considering the two traces $t_3 = \langle A, B, D, E, C \rangle$ and $t_4 = \langle A, B, E, D, C \rangle$, the subprocess $s1$ in Figure 4 is added to the net in Figure 1 to take care of the sequences of activities $\langle D, E \rangle$ and $\langle E, D \rangle$ that are inserted at the same place, i.e., after $B$. Moreover, according to whether the inserted actions represented by means of the subprocess are executed at most once, exactly once or more than once in $T$, a skipping transition is added to the net, the subprocess is added in sequence or it is nested in a loop block. In our example the subprocess is executed at most once and therefore a skipping transition that directly connects $B$ and $C$ is added to the net in Figure 4.

***Loop repair operations.*** In order to improve the simplicity of the repaired model, a special repair operation is dedicated to the identification of loops in the traces. The identification of a loop, whose body represents a behaviour already described in the model, allows the addition of a simple *loop back* transition instead of a new subprocess duplicating the behaviour already contained in the initial model. For example, given the net in Figure 1 and a trace $t_5 = \langle A, B, C, B, C \rangle$ the silent transition (loop back transition) in Figure 5 is added to the net, instead of a new subprocess accepting the second sequence $\langle B, C \rangle$.

***Remove unused part operations.*** In order to improve the precision and the simplicity of the repaired model $M'$, the parts of $M'$ that are no more used are removed, by aligning $T$ with $M'$ and detecting the parts of the model that do not contribute to the acceptance of a minimum number of traces.

In this paper we applied our technique on top of the results provided by the state-of-the-art ProM *Repair Model* plugin with the default configuration, which has been set by the authors to values providing the best results, according to their experiments [4].

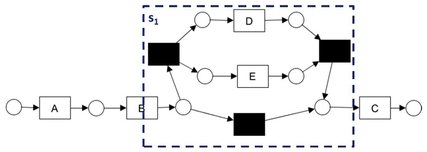

**Fig. 4.** An example of subprocess repair operation

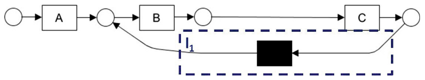

**Fig. 5.** An example of loop repair operation

## 3 Process Repair as a Multi-objective Optimization Problem

To repair a model, a set of changes $A$ (*repair operations*) are discovered and applied by the repair algorithm. Indeed, every subset $\bar{A} \subseteq A$ is able to partially fix the model $M$, so that a subset $\bar{T} \subseteq T$ of traces is accepted by the partially repaired model. Assuming that every operation $a \in A$ has a cost $c(a)$, we can formulate the problem of trading the number of traces accepted by the repaired model for the cost of repairing the model as a *multi-objective optimization problem* (MOP).

### 3.1 Multi-objective Optimization

In single-objective optimization, given a set $X$ of alternatives and a function $f : X \to \mathbb{Z}$, which returns a cost (benefit) value associated with each alternative in $X$, the single-objective optimization problem consists of finding an element $x^* \in X$ which minimizes (maximizes) $f$. Multiple objectives can be expressed through a finite set of functions $\{f_i : X \to \mathbb{Z} | i = 1...n\}$ defined on the set $X$. Solving the optimization problem intuitively requires to find elements in $X$ that give the best possible values for all the objective functions $f_i$ at the same time. It is often the case that functions $f_i$ assume their minimum/maximum in different points of $X$ so that there is not a single point in $X$ which simultaneously optimizes all $f_i$. For this reason the concept of *Pareto optimum* is introduced:

**Definition 1 (MOP).** Multi-objective Optimization Problem *(MOP) is defined by an $n$-tuple of functions $(f_1, f_2, ..., f_n)$ with $f_i : X \to \mathbb{Z}$ and a corresponding $n$-tuple of ordering operators on $\mathbb{Z}$ $(o_1, o_2, ..., o_n)$ where $o_i \in \{\leq, \geq\}$, $o'_i \in \{<, >\}$.*

**Definition 2 (Pareto optimum).** *A point $x^* \in X$ is a* Pareto optimum *for the MOP defined by $\langle (f_i), (o_i) \rangle$ if the following two conditions hold:*

– $\forall i \in \{1, ..., n\}$, $f_i(x^*)\ o_i\ f_i(x)$ *for all* $x \in X$,
– $\exists j \in \{1, ..., n\}$ *such that* $f_j(x^*)\ o'_j\ f_j(x)$, *for all* $x \in X$.

**Definition 3 (Pareto front).** *The image $F^* = \{(f_1(x^*), f_2(x^*), ...f_n(x^*) | x^* \in X^*\}$ of the set $X^*$ of points $x^*$ which are Pareto optima for the MOP defined by $\langle (f_i), (o_i) \rangle$ is called* Pareto front *for the MOP.*

Thus, a Pareto optimum provides a point that is equal or better than any other point for all the functions $f_i$ and it is better than any other point for at least one function $f_j$.

The Pareto front is a useful tool to describe the options that a decision maker has at disposal and to identify preferred alternative among the available ones. In particular, when problems with two objective functions are concerned, a graphical representation of alternative solutions can be obtained by drawing the Pareto front points on the Cartesian plane. The solutions (points) that are not on the Pareto front are by definition worse at least in one objective than the solutions on the front and so they can be ignored in the decision process.

## 3.2 Process Repair as a MOP

Process model repair can be seen as a MOP if the power set of the repair operations $\mathcal{P}(A)$ is taken as the space of alternative solutions $X$, i.e. $X = \{\bar{A}|\bar{A} \subseteq A\}$ and for every element in $X$, namely $\bar{A} \subseteq A$, the following are considered objective functions:

- **Number of accepted traces:** $N(\bar{A}) = |\bar{T}|$, the size of the set $\bar{T}$ of traces accepted by the model repaired by the operations in $\bar{A}$
- **Total Cost:** $C(\bar{A}) = \sum_{a \in \bar{A}} c(a)$, the sum of the costs for all repair operations in subset $\bar{A}$

where function $N(\bar{A})$ is to be maximized, while function $C(\bar{A})$ is to be minimized.

Having expressed process model repairs as a MOP, we can find a solution by following the approach in [12], which transforms the MOP into a satisfiability problem. The method consists of assuming a maximum value $\bar{C}$ for the total cost function $C$ and a minimum value $\bar{N}$ for the number of accepted traces function $N$, and writing a set of linear integer equations that describe the process repair problem under the constraints imposed by $\bar{C}$ and $\bar{N}$. Then, an SMT solver is used to find a solution or to establish that the problem is infeasible under constraints $\bar{C}, \bar{N}$. By varying $\bar{C}$ and $\bar{N}$ appropriately (e.g., incrementing $\bar{C}$ or decrementing $\bar{N}$ when no solution is found), the entire Pareto front can be precisely explored.

The first step for us is to translate the problem into a set of pseudo-boolean constraints (PBCs). A PBC is a formula involving booleans and linear integer arithmetics, having the form: $\sum_{i=1}^{n} a_i x_i \odot B$, where: $\odot \in \{<, \leq, = \neq, >, \geq\}$, $a_i, B \in \mathbb{Z}$, and all $x_i$ range over the set $\{0, 1\}$. Figure 6 shows a simple example of process model repair, including the repair operations, their costs and the traces accepted by the model repaired by each operation.

| Operation | Cost | Accepted Traces |
|---|---|---|
| $a_1$ | 3 | $t_1, t_2$ |
| $a_2$ | 2 | $t_1, t_3$ |
| $a_3$ | 4 | $t_2, t_3, t_4$ |

**Fig. 6.** An example of repair operations

The second step is to define, for a set $\bar{A} \subseteq A$ of repair operations, the vector $(\alpha_1, \alpha_2, ..., \alpha_{N_A})$ as the boolean-valued variables with the property $a_i \in \bar{B}$ iff $\alpha_i = 1$. In other words, $(\alpha_1, \alpha_2, ..., \alpha_{N_A})$ gives the characteristic function of $\bar{A}$. Similarly, for a set $\bar{T} \subseteq T$ of traces, $\bar{T}$ can be characterized by the vector $(\tau_1, \tau_2, ..., \tau_{N_T})$ of the boolean-valued variables with the property $t_i \in \bar{T}$ iff $\tau_i = 1$. To make the notation easier to read, we overload the semantics of variables $\alpha_i$ and $\tau_i$, making the assumption that when used in an integer context the boolean value *true* is interpreted as the integer value 1, *false* as 0.

The constraints on the objective functions "total cost" $C$ and "number of accepted traces" $N$ can be expressed as PBCs involving the variables $\alpha_j$ and $\tau_i$, respectively:

$$\sum_{i=1,\dots,N_A} c_i\alpha_i \le \bar{C} \qquad (1) \qquad\qquad \sum_{i=1,\dots,N_T} \tau_i \ge \bar{N} \qquad (2)$$

Let us define the matrix $\{m_{ij}\}$ with $i = 1, ..., N_T$ and $j = 1, ..., N_A$ such that $m_{ij} = 1$ if and only if trace $t_i$ requires the repair operation $a_j$ to be accepted by the repaired model. The following system of logical formulas model the relationship between repair operations and accepted traces:

$$\tau_i \leftrightarrow \bigwedge_{j=1,\dots,N_A \wedge m_{ij}=1} \alpha_j, \qquad \text{with } i = 1, ..., N_T \qquad (3)$$

Figure 7 shows the logical formulas for the example in Figure 6: the formula in the first row states that trace $t_1$ is accepted ($\tau_1$ is true) if and only if repair operations $a_1$ and $a_2$ are applied ($\alpha_1 \wedge \alpha_2$ are true). Similar conditions for $t_2$, $t_3$ and $t_4$ are shown in the remaining rows of the table.

$$\boxed{\begin{array}{l} \tau_1 \leftrightarrow \alpha_1 \wedge \alpha_2 \\ \tau_2 \leftrightarrow \alpha_1 \wedge \alpha_3 \\ \tau_3 \leftrightarrow \alpha_2 \wedge \alpha_3 \\ \tau_4 \leftrightarrow \alpha_3 \end{array}}$$

**Fig. 7.** Logical formulas for Figure 6

It can be proven that the set of formulas in Equation (3) is equivalent to the set of PBCs expressed by Equations (4a) and (4b), for all $i = 1, ..., N_T$. Figure 8 shows the set of PBCs obtained for the example in Figure 7.

$$\tau_i \ge 1 + \sum_{j=1,\dots,N_A} m_{ij}(\alpha_j - 1) \qquad (4a)$$

$$N_A\tau_i \le N_A + \sum_{j=1,\dots,N_A} m_{ij}(\alpha_j - 1) \qquad (4b)$$

With this transformation, all constraints that must be satisfied to solve our MOP problem are expressed in pseudo-boolean form. Specifically, the set of PBCs (1), (2), (4a), and (4b) defines the model repair problem $MRP = \langle\{c_j\}, \{m_{ij}\}, \bar{N}, \bar{C}\rangle$ of finding a subset of repair operations that are required to accept at least $\bar{N}$ traces with a repair cost not greater than $\bar{C}$, given the action costs $\{c_j\}$ and the relation between actions and traces specified by matrix $\{m_{i,j}\}$.

$$\boxed{\begin{array}{r} \tau_1 \ge 1 + (\alpha_1 - 1) + (\alpha_2 - 1) \\ \tau_2 \ge 1 + (\alpha_1 - 1) + (\alpha_3 - 1) \\ \tau_3 \ge 1 + (\alpha_2 - 1) + (\alpha_3 - 1) \\ \tau_4 \ge 1 + (\alpha_3 - 1) \qquad\qquad\quad \\ 3\tau_1 \le 3 + (\alpha_1 - 1) + (\alpha_2 - 1) \\ 3\tau_2 \le 3 + (\alpha_1 - 1) + (\alpha_3 - 1) \\ 3\tau_3 \le 3 + (\alpha_2 - 1) + (\alpha_3 - 1) \\ 3\tau_4 \le 3 + (\alpha_3 - 1) \qquad\qquad\quad \end{array}}$$

**Fig. 8.** PBCs expressing the relation between applied actions and accepted traces for Figure 6

The problem $MRP$ can be tackled by a Satisfiability Modulo Theory solver (`YICES`[2] was used in this work). If the problem

[2] YICES: `http://yices.csl.sri.com/`

turns out to be satisfiable, the accepted traces are identified by the true elements of $\{\tau_i\}$ and the required repair operations by the true elements of $\{\alpha_j\}$.

***Computing the Pareto front.*** The Pareto front for the model repair problem MRP can be computed using Algorithm 1. First (step 1), we compute the point $(C_T, N_T)$ of the front with maximum cost. Second (step 2), starting from $C = C_T$ and $N = N_T$, the maximum allowed cost $C$ is reduced by one and the maximum number $N$ of traces that can be accepted with that cost is searched, iteratively solving the problem $P(\boldsymbol{c}, \boldsymbol{m}, C, N)$ while decreasing $N$ until a satisfiable problem is found. When found, the point $(C, N)$ is added to the set $F$ and every point $(C', N')$ that is dominated by $(C, N)$ is removed from $F$; the cost is reduced by one and the loop is repeated. Upon exit, the algorithm returns the set of points in the Pareto front.

**Algorithm 1.** Computing the Pareto front for the Model Repair MOP

```
Input:
        $\boldsymbol{c} = c_1, ..., c_{N_A}$ vector containing the cost of repair operations,
        $\boldsymbol{m} = (m_{ij})_{i=1,...,N_T, j=1,...,N_A}$, matrix specifying what traces are
        repaired by what operations
Output:
        $F$, a set of points $(C, N)$ (cost, number of accepted traces),
        i.e. the Pareto front for the Model Repair MOP
// step 1: Compute the cost $C_T$ to have a model that accepts the whole set of traces
$C_T = \sum_{i=1,...,N_A}(c_i)$
add $(C_T, N_T)$ to $F$
// step 2: Follow the Pareto front
$C = C_T - 1$
$N = N_T$
while $C > 0$ do
   while $N > 0$ do
      if MRP problem $\langle \boldsymbol{c}, \boldsymbol{m}, C, N \rangle$ is satisfiable then
         add $(C, N)$ to $F$
         remove any $(C', N')$ dominated by $(C, N)$ from $F$
         break
      end if
      $N = N - 1$
   end while
   if $N = 0$ then
      break
   end if
   $C = C - 1$
end while
return $F$
```

## 4 Experimental Results

In order to evaluate the proposed multi-objective approach, we formulate the following research questions:

**RQ1** Does the Pareto front offer a wide and tunable set of solutions?
**RQ2** Does the Pareto front offer solutions that can be regarded as repaired models of good quality?

**RQ1** deals with the number and the variety of different solutions provided by *Multi-objective Repair*. In particular, the shape of the Pareto front and the number of the solutions in the Pareto front determine whether a wide range of alternatives that balance the two dimensions of interest is offered to business analysts. The Pareto front, in fact, might consist of points spread uniformly in the interesting region or it may be concentrated in limited, possibly uninteresting regions of the plane (e.g., near the totally repaired processes accepting almost all traces in $T$). In our specific setting the number of solutions available in the Pareto is dependent on the number of operations needed to repair the whole set of traces in $T$. In order to answer this research question, we look at the number of optimal solutions, as compared to the whole set of repair operations, and at the shape of the Pareto front.

**RQ2** investigates the quality of the repaired models in the Pareto front. Specifically, two important quality dimensions for repaired models [3] are taken into account: (i) *Precision*, i.e., how many new behaviours are introduced in the repaired model with respect to the real process being modelled; and, (ii) *Generality*, i.e., how many yet unobserved behaviours of the real process are accepted by the repaired model.

In the following, we report the case study, the metrics, the experimental procedure, and the results obtained to positively answer **RQ1** and **RQ2**.

### 4.1 Process under Analysis

The process used in the case study is a procedure carried out in the Italian Public Administration (PA). It deals with the awarding of public tenders by contracting administrations. Before the winners can be awarded with the final notification, the contracting administration has to verify whether the winners have all the necessary requirements. In detail, the procedure is activated when the tender reports and a temporary ranking are available. According to whether anomalous offers can be accepted or not, a further step of evaluation is required or the letters for winners, non-winners as well as the result letter can be directly prepared and entered into the system. At this point, the requirements of the temporary winners have to be verified. If such verification is successful, an award notice is prepared and officially communicated; otherwise, further clarifications are requested to the temporary winners and the verification is iterated. The award notice can be published through the Web, through the Council notice board or, if the reward is greater than a given threshold, it has to go to print.

A Petri net $M$ describing such public tender awarding procedure has been defined by a team of business experts as part of a local project. $M$ takes into account the "ideal" procedure described in official documents and is composed of 35 transitions, none of which silent, and 32 places. No concurrent behaviours and no routing transitions occur in $M$, while there are three alternative choices and a loop, involving 5 routing places[3]. Since discrepancies were found between $M$ and the actually observed execution traces $T$, a repaired model $M'$ was produced from $M$ using the ProM Repair Model plugin.

## 4.2 Metrics

In order to answer the above research questions, we use precision and generality metrics to compare $M'$ to a gold standard model $GSM$. Differently from the initial model $M$ which did take into account the generic "ideal" procedure described in official documents, the gold standard $GSM$ has been manually defined by a team of business analysts working on the real process of a specific institution. It contains all and only behaviours that are legal in the specific setting. Model $GSM$ contains 49 transitions and 38 places; it contains some parallel behaviours (2 routing transitions), several alternative paths and few loops (21 routing places). Transitions are decorated with transition probabilities estimated from real process executions.

***Precision.*** Precision ($P$) of a repaired model $M'$ measures the absence of extra-behaviour in $M'$ with respect to the behaviour it should contain. It is computed as the percentage of execution traces generated by the repaired model and accepted by the gold standard model $GSM$:

$$P(M') = \frac{|acc(GSM, T_{M'})|}{|T_{M'}|} \tag{5}$$

where $acc(M, T)$ is the subset of $T$ accepted by $M$ and $T_M$ is a set of traces stochastically generated by model $M$. It should be noticed that in the general case, where no $GSM$ is available, measuring the precision of a model might be quite difficult and might involve substantial manual effort. We expect that good models are characterized by high precision.

***Generality.*** *Generality* (G) measures the capability of the repaired model $M'$ to describe unobserved behaviours of the real process. We compute it as the percentage of traces generated by $GSM$ that are accepted by $M'$:

$$G(M') = \frac{|acc(M', T_{GSM})|}{|T_{GSM}|} \tag{6}$$

where $acc(M, T)$ is the subset of $T$ accepted by $M$ and $T_M$ is a set of traces generated by model $M$. We expect that good models are characterized by a high generality.

## 4.3 Experimental Procedure

The procedure followed in our experiments consists of the following steps:

[3] Detailed descriptions of the case study are available at the link http://selab.fbk.eu/mor/

1. **Trace generation.** Two sets of traces $T$ and $GT$ are generated from the gold standard model $GSM$ (in our experiments, $|T| = 100; |GT| = 10$). Each trace is generated by a random execution of the Petri net: at each step, the transition to fire is chosen according to the probabilities associated with the enabled transitions. The execution ends when no enabled transitions exist;
2. **Model Repair.** The set of traces $T$ is used to repair the initial model $M$, producing the set $A$ of operations required to fix it. For each operation $a \in A$, its cost $c(a)$, estimated as the number of transitions added by the repair operation $a$, and the set of traces $T(a)$ accepted by the repaired model due to the specific repair operation $a$ are stored;
3. **MRP Solver.** The Solver applies Algorithm 1 to obtain the *Pareto front*. Each point $P_i$ in the Pareto front is associated with a repaired model $M_i$;
4. **Compliance Computation for Generality.** The set of traces $GT$ is used to evaluate the generality of each repaired model $M_i$;
5. **Trace Generation from Repaired Models.** Each model $M_i$ is used to randomly generate a set $T_{M_i}$ ($|T_{M_i}| = 100$), using a uniform distribution of probabilities associated with the transitions;
6. **Compliance Computation for Precision.** Traces $T_{M_i}$ are checked against $GSM$ to measure the precision of model $M_i$.

Stochastic trace generation from $GSM$ and from the repaired models $M_i$ was repeated 10 times, to allow for the computation of average and standard deviation of the experimental metrics for precision and generality.

## 4.4 Results

Figure 9 shows the Pareto Front obtained by applying *Multi-objective Repair* to the presented case study. Each Pareto front point is associated with a model $M_i$, obtained by applying a different set of repair operations. The x-axis represents the cost of the repair operations applied to obtain model $M_i$, while the y-axis represents the number of traces in $T$ that are accepted by the repaired model $M_i$.

The shape of the Pareto front offers an approximately linear set of solutions that are quite well distributed along the two axes. There are 6 points in the central area of the plot, distributed two by two along the Pareto front. For each pair, the point with the lowest cost is clearly associated with a better solution since the more costly solution accepts just one additional trace. For example, $M_7$ (indicated with an arrow in Figure 9) and $M_8$ represent a pair of close points. A business analyst, in charge to choose between the two repaired models, would probably prefer $M_7$, since this solution presents a lower cost, sacrificing only one trace.

Considering that 16 different repair operations have been identified by the ProM repair plugin – hence, $2^{16}$ different sets of operations can be potentially built – the 12 solutions provided by *Multi-objective Repair* represent, for a business analyst in charge of repairing the initial model, a good selection of different trade-off solutions, all ensured to be Pareto-optimal. Manual inspection of the whole space of solutions would be unaffordable. Based on these considerations, we can answer **RQ1** positively.

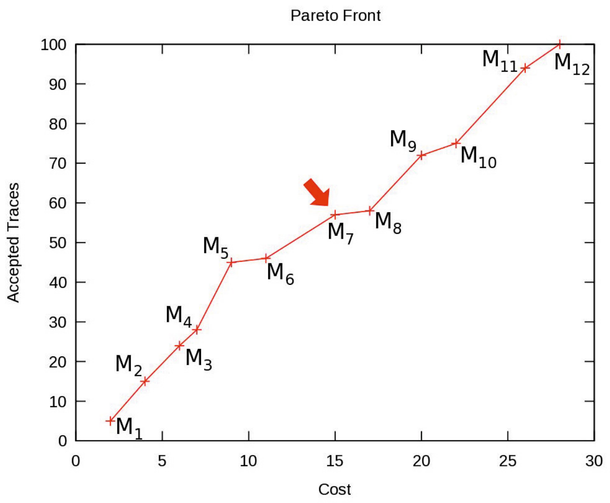

**Fig. 9.** Pareto Front obtained by applying *Multi-objective Repair* to the awarding of public tenders

Table 1 reports, for each repaired model $M_i$ in the Pareto front, the number of traces in $T$ accepted by $M_i$, the cost of the repair operations applied, and the values for precision and generality. Figure 10 plots the same data as a function of the repair cost. The low values for precision at increasing repair costs are due to the ProM repair algorithm, which tends to preserve old behaviours by introducing alternative silent transitions. These increase the number of extra-behaviours. As a consequence, the trend of the precision metrics is decreasing when the number of repair operations applied to the model grows.

The opposite trend can be noticed for the generality metrics (blue line in Figure 10). Starting from very low values for the poorly repaired models, the capability to reproduce new, unobserved behaviours increases together with the application of repair operations. It is worth noticing that in our case study the generality value for the repaired model accepting all traces in $T$, i.e., $M_{12}$, is exactly 1. In fact, the trace set $T$ used to repair the initial model $M$ provides exhaustive coverage of all possible process behaviours. Of course, this might not be true in the general case.

**Table 1.** Precision and generality for the models in the Pareto front

| | **# of accepted traces** | **Repair operation cost** | **Precision** | | **Generality** | |
|---|---|---|---|---|---|---|
| | | | Avg. | Std. dev. | Avg. | Std. dev. |
| $M_1$ | 5 | 2 | 1 | 0 | 0.04 | 0.07 |
| $M_2$ | 15 | 4 | 1 | 0 | 0.13 | 0.17 |
| $M_3$ | 24 | 6 | 1 | 0 | 0.22 | 0.08 |
| $M_4$ | 28 | 7 | 0.51 | 0.04 | 0.26 | 0.1 |
| $M_5$ | 45 | 9 | 0.53 | 0.07 | 0.41 | 0.08 |
| $M_6$ | 46 | 11 | 0.29 | 0.03 | 0.41 | 0.1 |
| $M_7$ | 57 | 15 | 0.51 | 0.04 | 0.51 | 0.11 |
| $M_8$ | 58 | 17 | 0.3 | 0.03 | 0.51 | 0.11 |
| $M_9$ | 72 | 20 | 0.4 | 0.03 | 0.66 | 0.1 |
| $M_{10}$ | 75 | 22 | 0.25 | 0.02 | 0.66 | 0.1 |
| $M_{11}$ | 94 | 26 | 0.26 | 0.03 | 0.98 | 0.04 |
| $M_{12}$ | 100 | 28 | 0.13 | 0.01 | 1 | 0 |

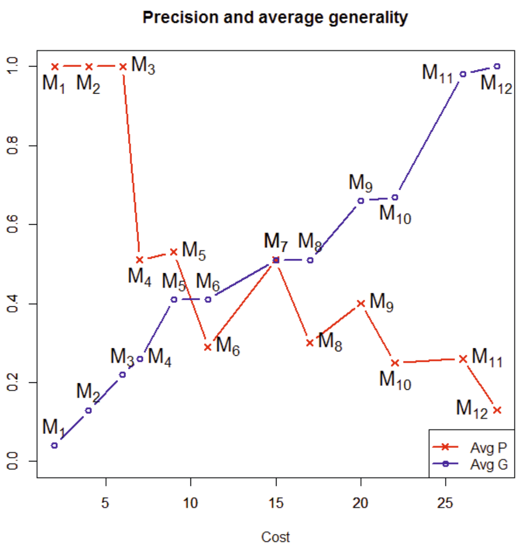

**Fig. 10.** Precision and average generality plots

The plot in Figure 10 shows that some of the intermediate solutions in the Pareto front (e.g., $M_5$, $M_7$ and $M_9$) offer quite interesting trade offs between precision and generality. For example, the repaired model $M_7$ is characterized by a precision and a generality of 0.51. At the same time, the additional complexity of this model in comparison with the initial model $M$ can be approximately measured by the repair cost (15), which is half of the total repair cost (28) for the fully repaired model $M_{12}$. If we can accept only half of the overall model complexity increase, we get approximately half precision and generality.

We can conclude that the Pareto front built by *Multi-objective Repair* provides business analysts with a set of tunable and good quality repaired models. The possibility to consider intermediate solutions (i.e., solutions in the central area of the Pareto plot) and to choose "how much" to repair the model (hence, how much to increase the model complexity), provides business analysts with a lot of flexibility in the trade-off between model quality and complexity. Based on these considerations, we can answer **RQ2** positively.

## 4.5 Discussion

We have manually inspected the repaired models in the Pareto front. We found that some cheap operations (e.g., the introduction of silent transitions, realizing loop back/skipping activities, or of small subprocesses) enable the acceptance of almost half of the traces in $T$ (see, e.g., $M_7$), at a cost that is around half of the total repair cost (28). Solutions located in the upper-right part of the Pareto, instead, are characterized by costly repair operations dealing with the acceptance of parallel behaviours.

The parallelization of activities and the management of mutually exclusive branches represent typical examples of challenging behaviours for repair techniques (in our case, for the approach implemented by the ProM plugin). The low precision values of some repaired models can be ascribed to these two types of criticalities. Concerning the first

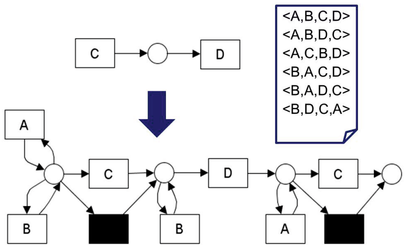

**Fig. 11.** Sequentialization of parallel activities

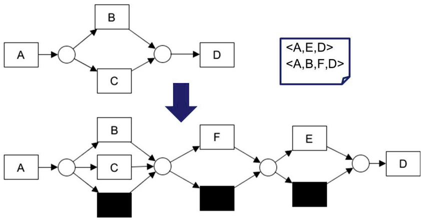

**Fig. 12.** Sequentialization of mutually exclusive activities

one (parallelization), indeed, the lack of a dedicated detector for parallel behaviours causes the insertion of subprocesses in charge of exploring the different interleavings of the parallel activities. Figure 11 shows a simplified view of this critical setting, which makes it also clear why extra-behaviours are introduced in the repaired model (e.g., $\langle C, B, D, C\rangle$). Similarly, Figure 12 shows a simplified representation of a particular case of the second criticality (mutually exclusive branches). When a new activity has to be added to a block of mutually exclusive branches, it is added in sequence at the join place as an optional activity, disregarding whether it is a new branch or part of an existing one. Figure 12 gives an idea of the extra-behaviour introduced in the repaired model (e.g., $\langle A, B, F, E\rangle$).

This analysis gives qualitative indications about the consequence of selecting a solution in the central area of the Pareto front (e.g., $M_5$ or $M_7$). A business analyst can repair the model at lower costs, while sacrificing execution traces involving more complex (and costly to repair) behaviours, such as parallel behaviours ($M_7$) or both mutually exclusive and parallel behaviours ($M_5$). The analysis provides also indications for the improvement of existing model repair algorithms, e.g., the need to introduce special rules dealing with parallelism and mutual exclusion.

### 4.6 Threats to Validity

Two main threats to validity can be identified in the presented case study, both related to the external validity, i.e., to the generalizability of the obtained results. The first threat concerns the investigation of a single case study. Results related to a single case study cannot be easily generalized. Nevertheless, the case study under analysis deals with a real procedure actually executed by Italian PA. The second threat is related to the specific repair tool and configuration used for identifying the set of repair operations. Different plugins and configurations would make the results more general. Nevertheless, the ProM plugin for process repair used in this work is among the most known in the literature.

## 5 Related Work

Reconciling execution information and process models, as done in process model discovery and repair, involves multiple, often contrasting, dimensions. Some works [5] apply post-processing analysis to simplify the discovered process models, while preserving the ability of the models to replay all the execution traces used for their generation. Others [9,3] use evolutionary algorithms to deal with these dimensions. De Medeiros et al. [9] apply a genetic algorithm to mine process models balancing the capability to reproduce behaviours traced in execution logs and extra-behaviours. Their algorithm optimizes a single-objective function, which combines under and over-generalization. A similar approach is taken by Buijs et al. [3], who use a genetic algorithm to discover a process model that not only balances under and over generalization, but also takes into account the model simplicity and generality, as well as the distance from an initial reference model. These works differ from ours because: (i) a new model is discovered rather than having an initial one repaired; and (ii) a single-objective function is used to combine all the dimensions to be optimized.

Multi-objective approaches have been applied to various fields of software engineering. For example, Harman et al. [13] present the first application of multi-objective optimization to the problem of test case selection. In their work, they study Pareto efficient genetic algorithms, such as NSGA-II, to maximize code coverage and minimize test cost during regression testing. Tonella et al. [11] introduce a multi-objective optimization algorithm to recover specification models that balance the amount of over- and under-approximation of application behaviours observed in traces. They show that multi-objective optimization performs outstandingly better than previous state-of-the-art approaches. Arito et al. [2] propose the formulation of the multi-objective problem of Test Suite Minimization (TSM) in terms of a set of pseudo-Boolean constraints. While our method adopts a similar formalization, the two considered problems differ in terms of involved constraints: in MRP a trace is accepted if and only if *all* associated repair operations are performed on the model, while in TSM each line of code can be executed by more than one test case, hence including *at least one* of them is enough to increase coverage.

Multi-objective optimization approaches have also been applied to the business process field, but never to optimize business process model repair. Marchetto et al. [8] apply a multi-objective technique to reduce intricate process models recovered from execution logs. In their work, two dimensions are investigated: complexity (measured in terms of analysts' understandability) and under-generalization. In Tomasi et al. [10], a third dimension is added to the two above: the business domain content of recovered processes. In both cases, the considered dimensions and the goal of the multi-objective optimization differ from the ones of this work.

## 6 Conclusions and Future Work

This paper presents a multi-objective approach to process model repair. It builds on top of state-of-the-art repair approaches and exploits the repair operations they provide to balance cost (in terms of complexity added to the recovered model) and advantages (in terms of traces accepted by the repaired model) of applying such operations to the initial model. Preliminary results, obtained by applying our approach to a real-life case study, indicate that: (i) the proposed *Multi-objective Repair* technique provides business analysts with a good selection of different solutions, all ensured to be Pareto-optimal; (ii) the returned solutions are tunable and good quality repaired models.

Future works will be devoted to performing further experiments involving larger case studies, as well as investigating the use of different configurations and tools for process model repair.

**Acknowledgments.** This work is partly funded by the European Union Seventh Framework Programme FP7-2013-NMP-ICT-FOF (RTD) under grant agreement 609190 - "Subject- Orientation for People-Centred Production".

## References

1. van der Aalst, W.M.P.: Business alignment: Using process mining as a tool for delta analysis and conformance testing. Requir. Eng. 10(3), 198–211 (2005)
2. Arito, F., Chicano, F., Alba, E.: On the application of SAT solvers to the test suite minimization problem. In: Proc. of the 4th Int. Symposium on Search Based Software Engineering (SSBSE), pp. 45–59 (2012)
3. Buijs, J.C.A.M., La Rosa, M., Reijers, H.A., van Dongen, B.F., van der Aalst, W.M.P.: Improving business process models using observed behavior. In: Cudre-Mauroux, P., Ceravolo, P., Gašević, D. (eds.) SIMPDA 2012. LNBIP, vol. 162, pp. 44–59. Springer, Heidelberg (2013)
4. Fahland, D., van der Aalst, W.M.P.: Repairing process models to reflect reality. In: Barros, A., Gal, A., Kindler, E. (eds.) BPM 2012. LNCS, vol. 7481, pp. 229–245. Springer, Heidelberg (2012)
5. Fahland, D., van der Aalst, W.M.P.: Simplifying mined process models: An approach based on unfoldings. In: Rinderle-Ma, S., Toumani, F., Wolf, K. (eds.) BPM 2011. LNCS, vol. 6896, pp. 362–378. Springer, Heidelberg (2011)
6. Gambini, M., La Rosa, M., Migliorini, S., Ter Hofstede, A.H.M.: Automated error correction of business process models. In: Rinderle-Ma, S., Toumani, F., Wolf, K. (eds.) BPM 2011. LNCS, vol. 6896, pp. 148–165. Springer, Heidelberg (2011)
7. Li, C., Reichert, M., Wombacher, A.: Discovering reference models by mining process variants using a heuristic approach. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 344–362. Springer, Heidelberg (2009)
8. Marchetto, A., Di Francescomarino, C., Tonella, P.: Optimizing the trade-off between complexity and conformance in process reduction. In: Cohen, M.B., Ó Cinnéide, M. (eds.) SSBSE 2011. LNCS, vol. 6956, pp. 158–172. Springer, Heidelberg (2011)
9. Medeiros, A.K.A.D., Weijters, A.J.M.M.: Genetic process mining: an experimental evaluation. Data Min. Knowl. Discov. 14 (2007)
10. Tomasi, A., Marchetto, A., Di Francescomarino, C.: Domain-driven reduction optimization of recovered business processes. In: Fraser, G., Teixeira de Souza, J. (eds.) SSBSE 2012. LNCS, vol. 7515, pp. 228–243. Springer, Heidelberg (2012)
11. Tonella, P., Marchetto, A., Nguyen, C.D., Jia, Y., Lakhotia, K., Harman, M.: Finding the optimal balance between over and under approximation of models inferred from execution logs. In: 2012 IEEE Fifth Int. Conf. on. Software Testing, Verification and Validation (ICST), pp. 21–30. IEEE (2012)
12. Van Veldhuizen, D.A., Lamont, G.B.: Multiobjective evolutionary algorithm test suites. In: Proc. of the 1999 ACM Symp. on Applied Computing, pp. 351–357. ACM (1999)
13. Yoo, S., Harman, M.: Pareto efficient multi-objective test case selection. In: Proc. of the 2007 Int. Symposium on Software Testing and Analysis, pp. 140–150. ACM (2007)