Optimizing the Trade-Off between Complexity and Conformance in Process Reduction

Alessandro Marchetto, Chiara Di Francescomarino, and Paolo Tonella

FBK-CIT, Trento, Italy
{marchetto,dfmchiara,tonella}@fbk.eu

Abstract. While models are recognized to be crucial for business process management, often no model is available at all or available models are not aligned with the actual process implementation. In these contexts, an appealing possibility is recovering the process model from the existing system. Several process recovery techniques have been proposed in the literature. However, the recovered processes are often complex, intricate and thus difficult to understand for business analysts.

In this paper, we propose a process reduction technique based on multi-objective optimization, which at the same time minimizes the process complexity and its non-conformances. This allows us to improve the process model understandability, while preserving its completeness with respect to the core business properties of the domain. We conducted a case study based on a real-life e-commerce system. Results indicate that by balancing complexity and conformance our technique produces understandable and meaningful reduced process models.

Keywords: Business Process Recovery, and Multi-Objective Optimization.

1 Introduction

Managing a business process and evolving the associated software system can be difficult without an accurate and faithful documentation (e.g., a model) of the underlying process. When such documentation is missing or inaccurate, it is possible to attempt to reconstruct it through process model recovery and mining.

Several works [1,2] in the literature propose techniques to recover process models starting from the analysis of different artifacts, such as the source code or execution logs. In our previous work [3], we used dynamic analysis to recover processes realized through Web systems.

The typical problems that are faced when process models are recovered from logs include: (i) the process size and complexity; (ii) under-generalization: models may describe only a subset of the actual system behaviors; and (iii) over-generalization: recovery algorithms may generalize the actual observations beyond the possible behaviors. The use of a large set of traces and of an appropriate algorithm can limit the impact of problems (ii) and (iii), but state of the art techniques have still a hard time with problem (i). In fact, process recovery and mining tools tend to produce process models that are quite difficult to understand, because they are overly complex and intricate (they are also called "spaghetti" processes [4]).

Existing approaches dealing with the process model size and complexity problem belong to two groups: clustering and frequency based filtering. Clustering takes advantage of the possibility of modularization offered by sub-processes [3]. Frequency-based process filtering removes process elements that are rarely executed [5]. Empirical evidence [6] shows that modularity affects positively process understandability, but its effect may be negligible due to factors such as the process domain, structure and complexity. Pruning processes according to the execution frequency [5] is useful to remove noise, but may lead to overly simplified processes, that do not capture the core business properties and activities. No existing process reduction technique treats process reduction as an intrinsically multi-objective optimization problem. Rather, they focus on a single dimension (e.g., process size) and take the others into account only implicitly. However, the problem is multi-dimensional: reduced processes are expected to be simpler to understand and manage (having lower size and complexity), but at the same time they should maximize their capability to represent all possible process flows, in a way that is meaningful and expressive for the business analysts.

In this work, we propose a process reduction technique which considers the multiple dimensions involved in this problem explicitly. We use multi-objective optimization to minimize at the same time the process model complexity and its non-conformances (i.e., inability to represent some execution traces). The proposed technique has been implemented in a tool and has been applied to a real-life e-commerce process implemented by a software system with a Web interface. Results indicate that by balancing both complexity and conformance we improve existing process reduction techniques.

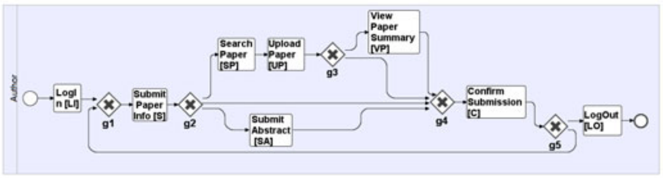
2 Business Process Reduction

Recovered process models are reduced by removing some process elements (e.g., sequence flows and activities), with the aim of limiting process size and complexity, while preserving a reasonable level of *conformance*, i.e., ability to fully describe the observed behaviors, enacted by the process under analysis.

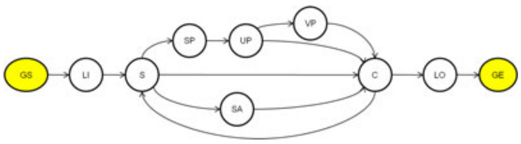
By *business process reduction* we mean a sequence of *atomic reduction operations*, each basically consisting of the removal of a direct connection between a pair of process activities, possibly followed by a cascade of further process element removals, occurring whenever an entire sub-process becomes unreachable from the start due to the initial removal. To perform business process reduction, we rely upon an intermediate process representation, called *activity graph*. The mapping between the original process model P and the activity graph AG_P must be invertible, since we want to be able to convert the reduced activity graph back to the original process.

Under the assumption that the initial BPMN process has only exclusive gateways[1] (as in case of processes recovered by our tool, see Section 5), we can safely remove them in the activity graph and reintroduce only those actually needed in the reduced process. Specifically, in the activity graph AG_P there will be a node n_a for each process activity a and an edge (n_a, n_b) for each pair of activities (a, b) such that there exists a path connecting them not traversing other activities. AG_P can be easily converted back to the original process P, since the exclusive gateways to be reintroduced are associated with nodes having fanout or fanin greater than 1. Though the number of gateways of the inverted and original process may differ, the semantics is however preserved and the original structure is sometimes obtainable by applying some gateway simplification rules. If the original BPMN process contains other types of gateways, in addition to the exclusive ones, it will not be possible to remove them from the activity graph. They will be explicitly represented by typed activity graph nodes, with the type used to distinguish different kinds of BPMN elements.

The example in Figure 1 represents the process for an on-line submission system. The corresponding activity graph contains a node for each activity/gateway of the process and two nodes (named GS and GE) representing the start and end event of the process.



(a) BPMN process model



(b) Activity graph

Fig. 1. Paper submission system: BPMN process and activity graph

Atomic Reduction. An atomic reduction consists of the removal of an edge from AG_P, followed by the removal of the subgraph unreachable from the start event. When the removed edge (n_a, n_b) dominates its target n_b (i.e., it belongs to all the paths from the start node GS to n_b), n_b becomes unreachable from the start event. We hence perform a reachability analysis from the start node GS to determine the subgraph of AG_P that is no longer reachable from GS due to the removal of (n_a, n_b). Such subgraph is also removed from the activity

[1] Gateways are used for managing control flow branching (both splitting and merging) in BPMN. According to their type (exclusive, inclusive, parallel and complex), they enable a different control flow behaviour.

graph, in order to ensure that the graph obtained from the atomic reduction is still a meaningful process representation. This means that the impact of a single atomic reduction may range from an individual edge removal to the removal of a (possibly large) subgraph of AG_P. It can be easily shown that the atomic reduction operation is commutative, i.e., the same activity graph is obtained by applying the atomic reduction for e_1 and for e_2 or in the opposite order.

In the activity graph in Figure 1, the removal of the edge (S, C) does not leave any subgraph unreachable from GS ((S, C) is not a dominator of C). In the BPMN process corresponding to the reduced activity graph, the only sequence flow that is removed is $(g2, g4)$. On the contrary, the removal of the edge (S, SA), a dominator of SA, makes node SA unreachable from the start node. By reachability analysis, we can easily determine that node SA as well as its outgoing edge (SA, C) must be also removed, since unreachable from GS. The corresponding BPMN process is shown in Figure 2. Not only the sequence flow $(g2, SA)$ has been removed, but also the process activity *Submit Abstract* as well as its outgoing sequence flow $(SA, g4)$.

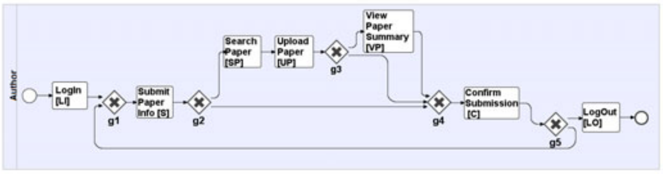


Fig. 2. Reduced process resulting after removing the edge (S, SA)

3 Process Complexity and Non-conformance

Three quality factors are dominant [7,8] concerning how to measure and evaluate the quality of (recovered) process models: *complexity*, *conformance* and *accuracy*. This work is focused on finding an optimal trade-off between the first two quality factors. Accuracy, which captures the ability of a process model to describe only feasible behaviors (i.e., executions admitted by the model should be also possible in reality), is not considered in this work for two reasons: (1) we apply process reduction to improve a single process mining technique, which has its own intrinsic accuracy level; (2) it is hard to get a precise and meaningful quantification of accuracy, since it is not possible to determine in an automated way which paths in the model are not possible in the real process implementation.

3.1 Process Complexity

The existing literature [7] proposes to measure the process complexity by resorting to the cyclomatic complexity, commonly used with software, and adapting it for processes. Hence, given a process model P, we measure its control-flow complexity $CFC(P)$ as follows:

$$CFC(P) = \sum_{g \in G(P) \wedge FOUT(g) > 1} FOUT(g) \tag{1}$$

where $G(P)$ is the set of all process control-flow elements (gateways, in BPMN) and $FOUT(g)$ is the number of the sequence flows outgoing from g (fanout). We take into account only decision points of the process, i.e., elements with fanout greater than one. A high value of CFC indicates a high number of alternative execution flows, thus denoting a process potentially difficult to read and understand for the analyst.

For example, the online submission process in Figure 1 contains five gateways, one with fanout 3, two with fanout 2 and two with fanout 1. The resulting process CFC is hence 7. For the reduced process in Figure 2, the CFC is 6.

3.2 Process Conformance

The conformance of a process model is its ability to reproduce the execution traces (i.e., the traces should be admitted by the process model) [8]. A trace is a sequence of events executed during the run of a process and stored in a log file. Table 1 shows some examples of execution traces for the submission process in Figure 1. For instance, trace t_3 represents an execution in which the user logs into the submission system (activity *LI*), submits the information about the paper (S), confirms the submission (C) and logs out (LO).

Since we aim for an optimal trade-off which minimizes complexity, we take the complement of conformance, i.e., non-conformance, as the metrics to be minimized. To compute the non-conformance $NConf(P)$ for process P, we interpret the model as a parser that can either accept or refuse a trace, regarded as its input string. A process model parses a trace if there is an execution flow in the process model that complies with the process model semantics and consists of the sequence of events in the trace. We assume that the parser associated with a process model is a *robust* parser, i.e., it has a parsing resume strategy which allows the parser to skip an input subsequence that cannot be accepted and to resume parsing from a successive input symbol. Specifically, if the trace consists of three parts $t =< p, m, s >$, such that the prefix p can be parsed and brings the process model execution to state S while the subsequence m cannot be accepted from state S, a resume strategy is a rule that transforms the process state S into S', such that the suffix s can be parsed from S'. Usually, the resume strategies implemented by parsers are heuristic rules to modify the state S by performing minimal changes to it, until an accepting state S' is reached.

Given a process model P and a trace $t \in T$ (the set of traces), we start parsing t with P in the start state. When event $e \in t$ is provided as input to process P (in the execution state S), if there does not exist any transition $S \longmapsto_e S'$ in P and there does not exist any resume rule R such that $S \longmapsto_R S' \longmapsto_e S''$, we recognize a non conformance $\langle S, e\rangle$, which is added to the non-conformance

Table 1. Example of execution traces

Trace	Event sequence	Trace	Event sequence
t_1	< LI, S, SP, UP, C, LO >	t_2	< LI, S, SA, C, LO >
t_3	< LI, S, C, LO >	t_4	< LI, S, SP, UP, VP, C, S, SP, UP, C, LO >
t_5	< LI, S, SP, UP, C, S, SA, C, LO >		

set. The size of the non-conformance set after trying to reproduce all the traces in T is our non-conformance metrics $NConf$. It should be noticed that if the same non-conformance $\langle S, e\rangle$ occurs multiple times (it appears in many traces) it contributes only as a $+1$ to $NConf$.

Under the assumption that the process model being considered contains only exclusive gateways (which is the case of the process mining tool used in our case study), the state of the process model execution when event e is provided as input consists of the last accepted activity. Only one parsing resume rule is needed for this class of process models. When the input activity b cannot be accepted by P, the preceding input activity a in the trace is considered. If the process model P contains the edge (a, b), parsing is resumed from state a. Computation of the $NConf$ metrics is also simplified for this class of process models. Specifically, the following formula can be used:

$$NConf(P) = \left| \bigcup_{t \in T} \{(a,b) | (a,b) \in t \wedge dc(a,b) \notin P\} \right| \tag{2}$$

where $dc(a, b)$ indicates the existence of a *direct connection* (i.e., an edge or a path containing only gateways as intermediate elements) between a and b in the process model P. Hence, we measure the number of unique transitions in the traces that are not reproducible in the process model. Let us consider the trace $t_2 = \langle LI, S, SA, C, LO\rangle$ from Table 1. The reduced process model in Figure 2 accepts the first two events, but then SA is not even present in the reduced process model. Thus, $(S, SA), (SA, C)$ are non-conformances. However, when LO is considered as input, parsing can restart from C according to the resume rule given above, such that (C, LO) is accepted. The final value of $NConf$ is 2 for this reduced process. It should be clear from this example that instead of parsing the string $\langle LI, S, SA, C, LO\rangle$, we can more easily obtain $NConf$ by just counting the number of consecutive pairs of events in the trace that do not have a direct connection in the reduced process model. $NConf(P)$ is 0 if the process model reproduces all traces, while a high value of $NConf(P)$ indicates that the process model is not able to reproduce many transitions in the traces.

If the considered process model P contains other types of gateways, in addition to the exclusive ones, we need to resort to a more general definition of $NConf(P)$ and, in particular, to a more complex parser strategy which implements the resume rules such as the one proposed by Rozinat et al. [8].

4 Multi-objective Optimization

We use multi-objective optimization to produce a reduced process model that minimizes both process complexity and non-conformance. Specifically, we rely on the Non-dominated Sorting Genetic Algorithm II (NSGA-II, Deb et al. [9]).

NSGA-II uses a set of genetic operators (i.e., crossover, mutation, selection) to iteratively evolve an initial population of candidate solutions (i.e., processes). The evolution is guided by an objective function (called fitness function) that evaluates the quality of each candidate solution along the considered dimensions (i.e., complexity and non-conformance). In each iteration, the Pareto front of the best alternative solutions is generated from the evolved population. The front contains the set of non-dominated solutions, i.e., those solutions that are not inferior to any other solution in *all* considered dimensions. Population evolution is iterated until a (predefined) maximum number of iterations is reached.

The obtained Pareto front represents the optimal trade-off between complexity and non-conformance determined by the algorithm. The business analyst can inspect the Pareto front to find the best compromise between having a model that conforms to the observed traces, but is quite complex vs. having a simpler, probably more understandable, but less adherent to reality, model.

Solution Encoding: In our instantiation of the algorithm, a candidate solution is an activity graph in which some edges are kept and some are removed. We represent such a solution by means of a standard binary encoding, i.e., a binary vector. The length of the vector is the number of edges in the activity graph extracted from the unreduced process model. While the binary vector for the unreduced process is entirely set to 1, for a reduced process each binary vector element is set to 1 when the associated activity graph edge is kept; it is set to 0 otherwise. For instance, the encoding of the unreduced submission process (Figure 1) consists of a vector of 13 elements (i.e., all the edges of the activity graph), all set to 1. In the vector encoding the reduced process in Figure 2, three bits (representing the edges *(S, C)*, *(S, SA)* and *(SA, C)*) are zero.

Initialization: We initialize the starting population in two ways, either randomly or resorting to the frequency-based edge-filtering heuristics. Random initialization consists of the generation of random binary vectors for the individuals in the initial population. The frequency based edge-filtering heuristics consists of removing (i.e., flipping from 1 to 0) the edges that have a frequency of occurrence in the traces below a given, user-defined threshold [5]. An initial set population has been obtained by assigning the frequency threshold (used to decide which edges to filter) all possible values between minimum and maximum frequencies of edges in the initial process.

Genetic Operators: NSGA-II resorts to three genetic operators for the evolution of the population: mutation, crossover and selection. As mutation operator, we used the bit-flip mutation: one randomly chosen bit of the solution is swapped. The adopted crossover operator is the one-point crossover: a pair of solutions is recombined by cutting the two binary vectors at a randomly chosen (intermediate) point and swapping the tails of the two cut vectors. The selection operator we used is binary tournament: two solutions are randomly chosen and the fitter of the two is the one that survives in the next population.

Fitness Functions: Our objective functions are the two metrics CFC and $NConf$, measuring the complexity and non-conformance of the reduced processes. For each candidate solution in the population, the reduced activity graph is obtained by applying the atomic reduction operations encoded as bits set to 0 in the corresponding binary vector. Then, the resulting reduced activity graph is converted back to a reduced process, evaluated in terms of CFC and $NConf$.

5 Case Study

We implemented our process recovery algorithm in a set of tools[2]: *JWebTracer* [3] traces the run of a Web application; *JBPRecovery* [3] infers a BPMN model; *JBPFreqReducer* implements the frequency-based heuristics *Fbr* to reduce processes by removing their rarely executed elements [5]; and *JBPEvo* implements the multi-objective genetic algorithm (MGA) presented in Section 4. In *JBPEvo*, the initial population can be randomly generated or it can be generated by applying the frequency-based heuristics Fbr.

We have applied our tools to the process model recovered from execution traces of the Web application Softslate Commerce[3]. Softslate is a Java shopping cart application that allows companies to manage their on-line stores. It implements functionalities to support the online retail of products (e.g., cart, order and payment) and to handle customer accounts, product shipping and administrative issues. The application consists of more than 200k lines of code written in Java/JSP. It uses several frameworks (e.g., Struts, Wsdl4j, Asm, SaaJ) and it can be interfaced with several database managers (e.g., MySql, Postgresql).

The aim of the case study was answering the following research questions investigating effectiveness and viability of the process reduction techniques in making the recovered processes more understandable and manageable.

- *RQ1*: *Does the shape of the Pareto fronts offer a set of solutions which includes a wide range of tunable trade-offs between complexity and conformance?* It addresses the variety of different, alternative solutions produced by *JBPEvo*. In particular, the shape of the Pareto front and the density of the solutions in the front determine the possibility to choose among a wide range of interesting alternatives vs. the availability of a restricted number of choices. The Pareto front, in fact, might consist of points spread uniformly in the interesting region of the CFC, $NConf$-plane, or it may be concentrated in limited, possibly uninteresting regions of the plane (e.g., near the unreduced process).
- *RQ2*: *Does the genetic algorithm improve the initial solutions (both random and edge-filtered)?* It deals with the margin of optimization left by the initial population, generated randomly or by Fbr. We want to understand if the improvements achieved through MGA are substantial.
- *RQ3*: *Are the reduced processes in the Pareto front understandable and meaningful for business analysts?* It deals with the quality of the solutions produced by MGA, as perceived by the business analysts. We want to understand the quality of the processes produced by MGA in terms of their understandability and meaningfulness.
- *RQ4*: *Do the processes obtained by applying multi-objective optimization offer qualitative improvements over those obtained by applying the edge-filtering heuristics?* It deals with the quality of the solutions produced by MGA compared to those obtainable by means of the Fbr heuristics.

[2] The tools are available at `http://selab.fbk.eu/marchetto/tools/rebpmn`
[3] `http://www.softslate.com/`

The procedure followed in the experimentation consists of: (1) *JWebTracer* has been used to trace 50 executions of Softslate Commerce (involving, on average, 12 user actions per execution). These executions exercise each application functionality at least once. (2) *JBPRecovery* has been used to analyze the recorded traces and build the unreduced BPMN process model. (3) *JBPFreqReducer* has been used to reduce the initial process model by applying the *Fbr* heuristics. A set of solutions has been obtained by varying the frequency threshold. The maximum population size considered is 100 processes. (4) *JBPEvo* has been run with two different types of initial populations, one randomly generated (MGA_R) and the other generated by *JBPFreqReducer* (MGA_F).

The dataset and recovered processes are available online[4]. Note that we address the research questions by resorting to a quantitative analysis when possible (specifically, for *RQ1* and *RQ2*). Other questions (*RQ3* and *RQ4*) involve more subjective evaluations, hence a qualitative analysis is more appropriate for them.

5.1 Quantitative Results

Figures 3, 4 and 5 show the Pareto fronts at different iterations of MGA_R, MGA_F and with the best solutions found by MGA_R, MGA_F and *Fbr* (highlighting the 9 processes selected for the qualitative analysis).
- **RQ1**. While for a small number of iterations MGA_R and MGA_F produce a Pareto front with few solutions (as shown by the shape of the fronts in Figures 3 and 4), after a large enough number of iterations (at least 10,000), the Pareto fronts present a smooth shape covering uniformly the CFC and $NConf$ ranges. After 1,000,000 iterations, the Pareto front of MGA_R has several solutions in the regions $CFC > 50$ and $NConf > 180$, while a smaller number of alternative trade-offs is offered (compared to MGA_F) in the region $CFC < 50$ and $NConf < 180$. In the region $CFC < 50$ and $NConf < 100$ MGA_F offers a lot of alternative solutions, while *Fbr* finds no solution at all. We conclude that MGA can produce a Pareto front which includes a wide range of tunable solutions. However, this requires a high enough number of iterations and a carefully initialized starting population (via *Fbr*). Otherwise, the choices offered to the user are potentially limited and sub-optimal.
- **RQ2**. The Pareto fronts obtained by iterating MGA_R show substantial improvements with respect to the initial population. The improvement is particularly impressive in the first 10,000 iterations, when the initial random solutions start to converge to a front, but major improvements are achieved when moving from 100,000 to 1,000,000 iterations. Some improvements with respect to the initial *Fbr* population have been observed also for MGA_F, but the margin for improvement left by *Fbr* (line for iteration 0) is lower than that left by the random initial population. Hence, convergence to the final Pareto front is achieved more quickly than MGA_R as confirmed by the time values[5] (in minutes) required by MGA to perform a given number of iterations reported in Table 2

[4] `http://selab.fbk.eu/marchetto/tools/rebpmn/opt_ext.zip`

[5] Running times for all the algorithms have been computed on a desktop PC with an Intel(R) Core(TM) 2 Duo CPU working at 2.66GHz and with 3GB of RAM memory.

(last column). After 10,000 iterations, in fact, only minor improvements can be obtained by further increasing the iteration number (e.g., 1,000,000). However, the most notable difference between Fbr and MGA_F is the density of the solutions in the Pareto front. In fact, such density is extremely increased in the front produced by MGA_F compared to Fbr, the former offering more alternative trade-offs. Hence, we can conclude that MGA improves both random and Fbr solutions, the latter to a lower degree, but with a substantially higher density of solutions available.

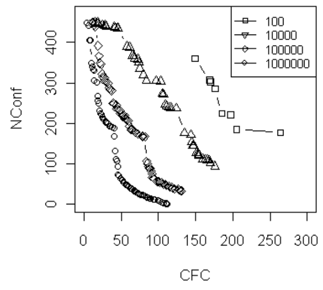


Fig. 3. Softslate: MGA_R Pareto fronts

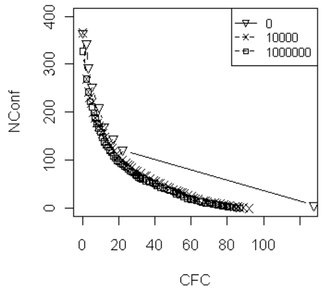


Fig. 4. Softslate: MGA_F Pareto fronts

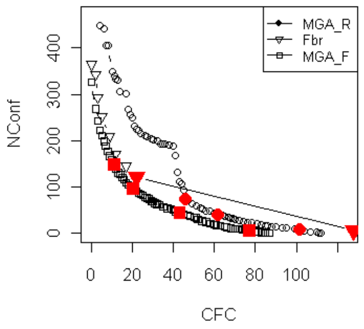


Fig. 5. Softslate: best Pareto fronts

Table 2. Unique processes and times in the Pareto fronts

Algo	Iter	Proc.	Unique Proc.	Time (min.)
MGA_R	100	9	8	1.18
MGA_R	10000	100	43	39.3
MGA_R	100000	100	68	294.2
MGA_R	1000000	100	67	2097.9
MGA_F	0	75	12	<0.2
MGA_F	100	83	14	1.24
MGA_F	10000	100	65	12.1
MGA_F	1000000	100	65	1077.2
Fbr	-	12	12	-

5.2 Qualitative Results

The points highlighted in Figure 5 represent the reduced processes randomly selected to be considered in the qualitative evaluation: *p4* and *p15* generated by Fbr; *p24*, *p25*, *p66* generated by MGA_R; and *p23* (Figure 6), *p27*, *p67*, *p41* generated by MGA_F. Table 3 provides some structural metrics (number of activities, gateways and sequence flows), as well as complexity and non-conformance metrics, for the unreduced process model and for the 9 selected processes.

In terms of difficulty of understanding, a big difference can be observed between reduced and unreduced processes, confirmed by the metrics shown in Table 3: the 9 selected processes have substantially lower values of structural and complexity metrics. However, this is paid in terms of non-conformance, as apparent also from Table 3. The point is whether such non-conformances affect minor aspects of the process (noise), or core activities and properties.

Table 3. Metrics for the 9 selected processes

Pr. id	#Act	#G	#SF	CFC	NConf
unreduced					
-	213	105	581	502	0
MGA_R					
p24	31	25	104	46	74
p25	39	37	125	62	40
p66	48	50	168	102	7
MGA_F					
p23	21	8	52	11	148
p27	23	15	60	20	97
p67	36	25	97	43	44
p41	46	38	142	77	6
Fbr					
p4	19	16	51	22	122
p15	46	44	188	128	6

Table 4. Some core business activities and properties

Business activities	
... a3.Add To Cart; ... a7.Clear This Cart; a8. Checkout; ... a10.Delete Item; a11.Log In; a12. Create New Account; ... a14.Confirm Order; ... a19.Reorder	
Business properties	
pr2	Remove has to follow Add To Cart, Pick Up or Reorder
pr3	After Edit it has to be possible to Edit Item or to Delete Item
pr5	Delete Item has to follow Add To Cart, Pick Up or Reorder
pr15	After Pick Up it has to be possible to choose among Edit, Remove, Continue Shopping, Save This Cart, Clear This Cart and Checkout
pr19	Confirm Order has to follow Checkout
pr21	Checkout has to be executed after the Log In or the Creation of a New Account; otherwise it has to be possible to Log In or to Create a New Account immediately after the Checkout
. . .	

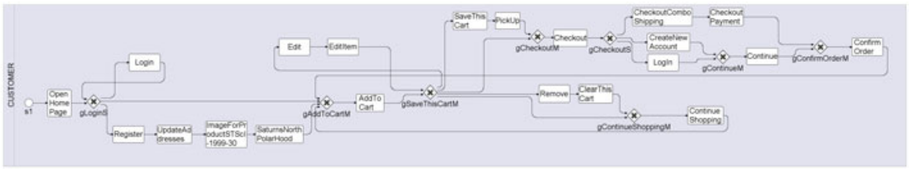


Fig. 6. Softslate: recovered reduced process

In order to assess the quality of the 9 selected processes from the point of view of the business analyst, we have identified 19 core business activities and 28 core business properties that pertain to the domain being modeled (Table 4 shows some examples). We assume that a good process model should represent explicitly all core business activities and should encode a process dynamics that complies with the core business properties. Core activities and properties have been defined by one of the authors not involved in the process recovery phase and without any knowledge about the recovered models (see Table 5). For instance, *Login* and *Checkout* (activities *a11* and *a8*) are core business activities, i.e., activities that any analyst would expect to be modeled explicitly. While an example of a core business property is: the user must be logged-in in order to be allowed to complete the Checkout operation (property *pr21*).

Table 5. Activities and properties modeled in the reduced processes

unreduced	MGA_R			Fbr		MGA_F			
	p24	p25	p66	p4	p15	p23	p27	p41	p67
Business Activities									
19	17	18	19	15	19	15	16	19	18
Business Properties									
19	19	21	25	18	24	19	22	25	25

- **RQ3**. Most of the 19 core activities are modeled in all 9 reduced processes (see Table 5). The highest number of missing core activities is 4 and is associated with the two processes (*p4* and *p23*) having among the lowest CFC values (22 and 11, respectively). The process *p27* has also low complexity ($CFC = 20$) and it misses 3 core activities (*a2*, *a16* and *a19*), missing in *p4* and *p23* too.

With regards to the business properties, something unexpected happened. Reduced processes tend to enforce the same number or more business properties than the initial, unreduced process recovered from the traces (note that the unreduced process covers 19 out of 28 identified properties). As shown in Table 5, there is only one exception: process *p4*, which enforces 1 property less than the unreduced process. Processes *p66*, *p41* and *p67* generated by MGA enforce 6 more core business properties than the unreduced process. Process *p15*, generated by Fbr, enforces 5 more properties. The explanation for this unexpected improvement is that the unreduced process was obtained by generalizing the behaviors observed in the execution traces, which contain noise and irrelevant details. This resulted in over-generalization of the possible execution paths, giving raise to extra behaviors, incompatible with the business domain. For instance, in the unreduced process we can find the following path: "⟨*Add To Cart, ..., Continue Shopping, Create New Account, ..., Confirm Order*⟩", which has never been traced in the logs and is forbidden by the Web application (core property *pr19*: "Confirm Order has to follow Checkout"). It appears as a consequence of the many complex paths combined together in a single model.

Moreover, it is possible to observe that even when properties do hold in the unreduced process, the process is so complex and intricate, that they can be hardly verified by humans, thus potentially becoming not much meaningful for the analysts. For example, while it is feasible for an analyst to check the failure of property *pr2* for process *p23* (see Figure 6), it is practically impossible for him to verify that the same property is true in the unreduced process.

We investigated also the reasons why none of the considered reduced processes realizes all 28 core business properties. We discovered three main problems: (1) one or more business activities involved in the property have been removed by the reduction algorithm (e.g., removal of activity *a10* from *p23* causes the failure of *pr3* and *pr5*); (2) a sequence flow that would allow to verify the business property has been removed by the reduction algorithm (e.g., removal of the sequence flow between the activities *Add To Cart* and *Clear This Cart* causes the failure of *pr15*); (3) over-generalization, which, as explained above, makes the unreduced process capable of enforcing only 19 out of 28 core business properties (e.g., the sequence of activities ⟨*Add To Cart, Continue Shopping, Continue, ...*⟩, possible in *p41*, is non-allowed by Softslate, as indicated also by property *pr18*). Cases 1 and 2 are instances of under-generalization, while 3 is the case of an over-generalized model. We analyzed the trend of under and over-generalization problems for the 9 selected processes, at decreasing complexity and increasing non-conformance (see Table 3). In these 9 cases, under-generalization problems grow at decreasing complexity (they are strictly related with the level of non-conformance), while over-generalization problems grow at increasing complexity, due to the excessive number of paths in the model.

Summarizing, a small number of business activities and properties are missing in the reduced processes, which make them meaningful to business analysts, especially in comparison to the unreduced process. Moreover, process reduction is almost always beneficial to the modeling of the core business properties, thanks to its implicit capability of reducing over-generalization.

- **RQ4**. To qualitatively compare the results achieved by *Fbr* and MGA_F, we consider the processes generated by the two algorithms that are within similar ranges of *CFC* and *NConf*: *p4* vs. *p23* and *p27*; *p15* vs. *p67* and *p41*. At lower complexity, the processes produced by MGA_F (*p23* and *p27*) include the same or more business activities. They always comply with more business properties (*p23* enforces 1 property more than *p4*; *p27* 4 properties more). While process *p23* produced by MGA_F is comparable to (actually, slightly better than) *p4* in terms of modeled core business activities and properties, it has half the *CFC* complexity of *p4*. This clearly indicates that the multi-objective optimization performed by *MGA* is extremely effective in determining a trade-off able to keep complexity low (11 vs. 22), while at the same time minimizing the non-conformances, which allows the process model to be comparatively similar to the more complex process *p4*, obtained by means of frequency-based reduction. The benefits of multi-objective optimization are apparent when *p4* and *p27* are compared. In this case, the *CFC* complexity is similar (22 vs. 20, respectively), but process *p27* has much less non-conformances (97 vs. 122), as a result of the multi-objective optimization. This can be appreciated in terms of properties that are meaningful for the business analyst. In fact, *p27* models 4 core business properties more than *p4* (22 vs. 18) and 1 more business activity.

When comparing *p15* vs. *p67* and *p41*, the number of business activities and properties is similar (differing by at most ± 1). However, the process models produced by *MGA* are far less complex, with *CFC* respectively equal to 43 and 77, vs. *p15* having *CFC* equal to 128. In the first case (process *p67*), multi-objective optimization is able to reduce the complexity of the model by a factor of around 1/3 at almost unchanged core business activities and properties. In the second case (process *p15*), the exact same amount of non-conformance (namely, 6) is achieved by a process model having almost half complexity (77 vs. 128).

In conclusion, multi-objective optimization is overall effective in optimizing the trade-off reducing the complexity of the model while leaving unchanged the business properties captured by the model and its non-conformance, hence representing a substantial improvement over the models obtainable by means of *Fbr*. As remarked above, *MGA* offers also a wider range of alternative solutions.

5.3 Threats to Validity

It is always hard to generalize the results obtained on a single case study. The selected application, however, seems to be representative of the e-commerce domain, where Web applications are used to implement the trading processes.

Two threats to validity impact the procedure of the case study. The first concerns the used executions traces. Since different models can be recovered considering different sets of traces, we applied the functional coverage criterion for the selection of application executions to be used. The second threat concerns the stochastic nature of the *MGA* algorithm. We considered different runs of the algorithm and obtaining comparable results, we detailed only one of them.

Finally, the qualitative analysis could have been influenced by the limited number of processes analyzed and the strong subjectivity characterizing the analysis. To limit this latter threat, the person in charge of identifying the business information was not involved in the process recovery phase.

6 Related Works

Process recovery deals with the analysis of a process implementation to extract models of the underlying process. For instance, Zou et al. [2] presented a two-step approach to recover business processes from Web applications. Process mining techniques (e.g., [10], [5]) try to infer processes by analyzing the workflow logs, containing information about organizations and process executions. In this area, most of the effort has been devoted to control flow mining (e.g., see the α algorithm proposed by van der Aalst et al. [5]).

Mining techniques often produce large and intricate processes. Assuming that this is due to spurious information contained in the traces, some works prune the model by removing process elements having low execution frequencies [5]. In our case study, we made a direct comparison with the frequency-based process reduction, showing the superiority of our approach in reducing process complexity and non-conformance. Other works cluster segments of traces and mine a number of smaller process models, one for each different cluster [10]. We used clustering [3] to modularize the processes into sub-processes for improving their understandability. However, the effects of modularity can be limited by the process domain, structure and complexity [6]. Process reduction, hence, is applied after process recovery and after the application of existing techniques for reducing process complexity (e.g., noise pruning, modularization).

Some works propose the use of evolutionary algorithms to balance under and over-generalization when mining a process model. For instance, Alves de Medeiro et al. [11] apply a genetic algorithm to mine a process balancing the ability of reproducing behaviors traced in logs and extra-behaviors. Their algorithm optimizes a single-objective function which combines under and over-generalization. On the contrary, we aim at two objectives at the same time, complexity and non-conformance. The former (absent in [11]) descends from the analysts' understandability needs, while our notion of non-conformance corresponds to their of under-generalization.

7 Conclusion

We have presented a multi-objective process reduction technique for reducing large processes by balancing complexity and conformance. We conducted a case study involving a real-life process exposed as a Web application. Results indicate that: (1) *MGA* produces a rich, fine grained, evenly distributed set of alternatives; (2) *MGA* improves state of the art reduction techniques; (3) convergence of *MGA* depends on the choice of the initial population; (4) though reduced, processes produced by *MGA* include relevant business activities and properties (i.e., we deem them as meaningful for analysts); (5) *MGA* produces solutions that are optimal along multiple dimensions at the same time. Future works will be devoted to perform further experiments, involving additional case studies and including other process mining tools.

References

1. van der Aalst, W., Weijter, A., Maruster, L.: Workflow mining: Discovering process models from event logs. IEEE Transactions on Knowledge and Data Engineering 2004(16) (2003)
2. Zou, Y., Guo, J., Foo, K.C., Hung, M.: Recovering business processes from business applications. Journal of Software Maintenance and Evolution: Research and Practice 21(5), 315–348 (2009)
3. Di Francescomarino, C., Marchetto, A., Tonella, P.: Cluster-based modularization of processes recovered from web applications. Journal of Software Maintenance and Evolution: Research and Practice (to appear) doi: 10.1002/smr.518
4. Veiga, G.M., Ferreira, D.R.: Understanding spaghetti models with sequence clustering for prom. In: Proc. of Workshop on Business Process Intelligence (BPI), Ulm, Germany (2009)
5. van der Aalst, W., van Dongen, B., Herbst, J., Maruster, L., Schimm, G., Weijters, A.: Workflow mining: A survey of issues and approaches. Journal of Data and Knowledge Engineering 47(2), 237–267 (2003)
6. Reijers, H.A., Mendling, J.: Modularity in process models: Review and effects. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 20–35. Springer, Heidelberg (2008)
7. Cardoso, J., Mendling, J., Neumann, G., Reijers, H.: A discourse on complexity of process models. In: Proc. of Workshop on Business Process Intelligence (BPI), Australia, pp. 115–126 (2006)
8. Rozinat, A., van der Aalst, W.: Conformance checking of processes based on monitoring real behavior. Information Systems 33(1), 64–95 (2008)
9. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
10. Bose, R., van der Aalst, W.: Context aware trace clustering: Towards improving process mining results. In: Proc. of Symposium on Discrete Algorithms (SDM-SIAM), USA, pp. 401–412 (2009)
11. Alves de Medeiros, A., Weijters, A., van der Aalst, W.: Genetic process mining: An experimental evaluation. Journal of Data Mining and Knowledge Discovery 14(2), 245–304 (2006)