Discovering and Navigating a Collection of Process Models using Multiple Quality Dimensions

J.C.A.M. Buijs, B.F. van Dongen, and W.M.P. van der Aalst

Eindhoven University of Technology, The Netherlands
{j.c.a.m.buijs,b.f.v.dongen,w.m.p.v.d.aalst}@tue.nl

Abstract. Process discovery algorithms typically aim at discovering a process model from an event log that best describes the recorded behavior. However, multiple quality dimensions can be used to evaluate a process model. In previous work we showed that there often is not one single process model that describes the observed behavior best in all quality dimensions. Therefore, we present an extension to our flexible ETM algorithm that does not result in a single best process model but in a collection of *mutually non-dominating* process models. This is achieved by constructing a Pareto front of process models. We show by applying our approach on a real life event log that the resulting collection of process models indeed contains several good candidates. Furthermore, by presenting a collection of process models, we show that it allows the user to investigate the different trade-offs between different quality dimensions.

Key words: process mining, process model quality, process model collection

1 Introduction

The goal of *process discovery* in process mining is to automatically discover process models that accurately describe processes by considering only an organization's records of its operational processes [1]. Such records are typically captured in the form of *event logs*, consisting of cases and events related to these cases. Over the last decade, many such process discovery techniques have been developed [1, 5], producing process models in various forms, such as Petri nets, BPMN models, EPCs, YAWL models etc. Furthermore, many authors have compared these techniques by focussing on the properties of the models produced, while at the same time the applicability of various techniques have been compared in case-studies. However, currently no algorithm produces a collection of process models for the user to choose from. Therefore, in this work we extend our genetic discovery algorithm to construct not a single, but a collection of process models. By using the notion of a Pareto front, only the best process models are kept that show different trade-offs between the quality dimensions.

Four quality dimensions are generally used to discuss the results of process discovery [1, 5], namely:

Replay fitness quantifies the extent to which the discovered model can accurately reproduce the cases recorded in the log.
Precision quantifies the fraction of the behavior allowed by the model which is not seen in the event log.
Generalization assesses the extent to which the resulting model will be able to reproduce future (unseen) behavior of the process.
Simplicity quantifies the complexity of a process model (e.g. number of nodes).

Surprisingly, many process discovery algorithms only focus on one or two of these dimensions [5]. Therefore, we proposed the ETM-algorithm (which stands for *Evolutionary Tree Miner*) in [5, 6] to seamlessly include different quality dimensions in the discovery process. However, until now, weights needed to be given to the different dimensions. Although assigning weights is a common way of aggregating multiple fitness values to a single one, there are some disadvantages. For instance beforehand the impact of a change in the process model on the value in a particular dimension is not known. This makes assigning weights to measures difficult, especially since the sensitivity of each measurement to changes in the process model is different.

The remainder of the paper is structured as follows. Next, in Section 2, the Pareto front is explained in more detail and common ways to construct such a Pareto front of candidates are discussed. Then, in Section 3, we briefly explain the ETM-algorithm and how it has been extended to build a Pareto front of mutually non-dominating process models. Section 4 then presents a case study where the number of edits from a reference process model is used as a fifth quality dimension. Section 5 concludes the paper.

2 Multi-Objective Optimization

Optimizing multiple objectives at the same time is a common challenge in optimization problems [7]. One of the simplest solutions is to use a weighted average over all the quality dimensions to produce a single quality measure. However, this method has several drawbacks:

1. Determining the correct weights upfront is difficult: structural changes on the candidates have unknown effects on the value for a dimension.
2. Values need to be normalized for comparison: a common way to fix the previous issue is by normalizing the values. However, dimensions can still respond differently to changes. Furthermore, a normalized value often provides less information than the absolute value.
3. Only one solution is provided: only the candidate with the best weighted average is presented. However, no insights in the different trade-offs among the dimensions is provided to the user.

The so-called *Pareto* front is often used as an alternative to the weighted sum [7, 16]. The general idea of a Pareto front is that all members are *mutually*

non-dominating. A member dominates another member if for all quality dimensions it is at least equal or better and for one strictly better, than the dominated member. Since all members in the Pareto front are mutually non-dominating (neither of them dominates another member) they represent different trade-offs in the quality dimensions sacrificing on one quality dimension to improve on another. This concept was originally proposed by Vilfredo Pareto to explain economic trade-offs [11].

An example of a Pareto front in two dimensions is shown in Fig. 1. Each dot in the graph represents a process model with a certain replay fitness and precision value, the two quality dimensions used for this Pareto front. For each dimension a bigger value indicates a better candidate, e.g. the goal is to obtain a process model in the top right corner of the chart. However, often there is no single model that is able to score perfectly on all dimensions. The open dots in the lower middle area of Fig. 1 are non-optimal process models, e.g. one of the dimensions can be improved without reducing the quality in (any of) the other dimension. The closed black dots represent the current estimation of the Pareto front. For these process models there is currently no model known where one dimension has a better score without reducing one of the other dimensions. The bigger dots show the seven most diverse process models in the current front, which can be used to truncate the Pareto front by keeping only one representative for a group of similar process models. The ideal or real Pareto front, as indicated by the curved line, shows that some improvements can still be made.

The construction of a Pareto front has been frequently applied in evolutionary algorithms. As described in [7], an evolutionary multi-objective optimization (EMO) algorithm aims at both converging close to the real, yet unknown, Pareto front while at the same time maintain a good diversity among the candidates

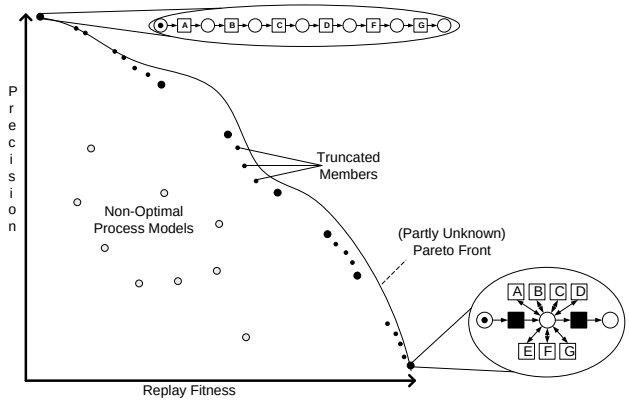


Fig. 1: Pareto Front of the two quality dimensions replay fitness and precision. The hollow dots are non-optimal process models, the small black dots are discovered process models representing the current Pareto front and the big black dots are the 7 most diverse among the currently known process models.

on the current Pareto front. The two most common, state-of-the-art, evolutionary algorithms that build a Pareto front of candidates are the NSGA-II [8] and SPEA2 [18] EMO algorithms. Between the two there is a slight but important difference in the way the fitness evaluation is performed and how the Pareto front is truncated. In short, SPEA2 calculates the fitness using both the dominance and the density of the population. The dominance includes the number of individuals dominated by that candidate. But also the number of individuals dominating the candidate. Furthermore, dominators are given a weight by using the total number of candidates they dominate. The density of the whole population is obtained by first calculating the distances between all candidates, and then considering the distance between a candidate and the k closest candidate. The NSGA-II algorithm defines the fitness of a candidate by the 'non-domination rank' (e.g. the number of candidates dominating the current candidate, candidates with a non-domination rank of 0 are on the 'actual' Pareto front). NSGA-II also has the notion of (crowding) distance which they calculate by taking the distance to the candidate better and the candidate worse in each dimension. These distances are normalized by the total value range of that dimension. The overall crowding distance of a candidate is the average over all dimensions. However, candidates that are at the extremes of a particular dimension, get very low (good) distance values. In this way the extreme candidates are always maintained. Both approaches select the candidates for the new population of the next generation according to a binary tournament using the fitness assignment. They also allow for truncation of the Pareto front using the same fitness function. NSGA-II and SPEA2 have been extensively compared regarding performance but no clear overall winner can be announced [4, 10, 13]. However, for our particular situation the crowding distance of the NSGA-II algorithm is chosen. The most important reason is the fact that extremes in each dimension are maintained, providing the user with extreme examples.

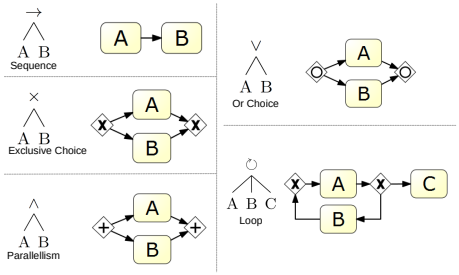


Fig. 2: Process trees operators and their block-structured BPMN translation.

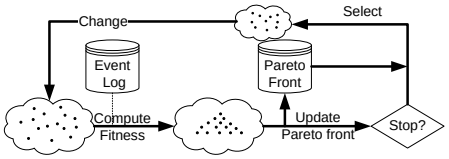


Fig. 3: The different phases of the genetic algorithm.

3 Extending the ETM to Discover a Pareto Front

The ETM (*Evolutionary Tree Miner*) is a flexible evolutionary process discovery algorithm [5]. The ETM-algorithm is able to discover tree-like process models that are sound and block-structured. Examples of the different process tree constructs are shown in Fig. 2. Overall the ETM-algorithm follows the genetic process shown in Fig. 3. The input of the algorithm is an event log describing the observed behavior and, optionally, one or more reference process models. First, different quality dimensions for each candidate in the current population are calculated. Then, for each new candidate, the Pareto front is updated. In the next step, certain stop criteria are tested such as exceeding a time limit or whether the user canceled execution. If none of the stop criteria are satisfied, a new selection of candidates from the Pareto front is made, which forms the new population. The candidates in this population are then changed, the fitness for each of the quality dimensions is again calculated and they are presented to the Pareto front for possible inclusion. This process is continued until at least one stop criterion is satisfied and the current Pareto front is then returned. More details on the ETM-algorithm and the process tree notation used can be found in [5, 6].

The ETM-algorithm is able to incorporate different quality dimensions. In general the four standard process discovery quality dimensions of replay fitness, precision, generalization and simplicity are included. However, the ETM-algorithm is also able to incorporate other quality dimensions during discovery. Examples are the conformance of a process model to a given set of rules [3,9,14], the predicted risk [12, 15] of a process model, the predicted cost for handling a case with the given process model [17], the overall performance of the process model [2], etc. As long as a quality dimension can be calculated by considering the process model and possibly the event log, and can be influenced by changing the process model, it is valid for inclusion in the quality evaluation of a process discovery algorithm.

In the original ETM-algorithm the weights for each quality dimension, as provided by the user, are used to calculate a single fitness value per candidate and sort them accordingly. Furthermore, when the ETM-algorithm terminates, only the best candidate is returned. In this work the ETM-algorithm is extended with a Pareto front cache that maintains the current Pareto front during the different generations of the ETM-algorithm. At the end of each generation the currently evolved and evaluated population is added to the Pareto front, if they are not dominated by any element currently in the Pareto front. At the beginning of the next iteration a fixed number of candidates is selected from the Pareto front, since the front can grow larger than the desired population size.

In order to select the best candidate from the Pareto front for the population/input of the new generation, a fitness value is calculated. Here we use a fitness calculation inspired by the crowding distance used in the NSGA-II [8] algorithm, as was discussed in Sec. 2. This fitness metric consists of two parts: calculating the number of candidates that dominate the current candidate and calculating the crowding distance of the candidate. The first part of the metric

results in an integer value, namely the number of candidates in both the Pareto front and the current population that dominate the particular candidate. The second part of the metric results in a value between 1 and 0 inclusive and represents the 'crowding distance' of a candidate. A value close to 0 indicates that the candidate is not in a crowded area, and a value of 0 indicates the candidate is at one of the extremes for at least one dimension. The crowding distance is calculated per dimension by considering the distance of a candidate with the next candidate that is worse and the next that is better. By normalizing this distance by the overall distance between the worst and best candidate of that dimension, a relative value is obtained. It is important however to assign boundary solutions, e.g. solutions that are best or worst in at least one dimension, a good crowding distance (e.g. a low value) to ensure that they are kept in the Pareto front. The crowding distance thus favors candidates that are diverse, for instance during the selection of candidates for mutation. The ETM with the Pareto front extension is implemented in the ProM framework[1].

4 Application on a Real Life Event Log

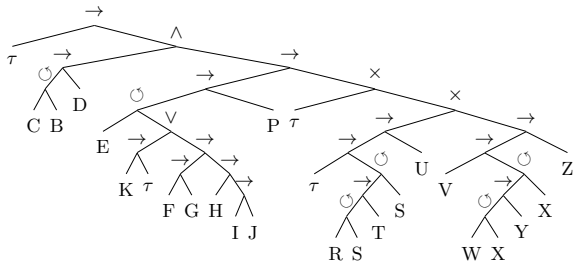
In this section we demonstrate our Pareto extended approach using the four quality dimensions of replay fitness, precision, generalization and simplicity. Additionally, we use the edit distance quality dimension as a fifth dimension to evaluate the similarity to a given process model [6]. The ETM-algorithm is started from a given process model and keeps track of the number of edits (add, remove, or update of a single node) made to this model. This allows the user to select a process model that is more or less similar to the reference process model that was provided. We apply the ETM-algorithm as presented in [6] on the same data set used there, but now with our Pareto extension.

The input of the ETM-algorithm is an event log describing the processing of building permits within a municipality. This event log comes from one of the municipalities participating in the CoSeLoG project[2]. The event log contains $1,434$ cases with $8,577$ events in total and 27 different event classes or activities. The provided reference model, as is shown in Fig. 4, is very detailed with many checks that the employees in practice did not always perform (usually with good reasons). Therefore, the municipality is interested to know where the main deviations are with respect to the reference process model.

The ETM-algorithm was run for $5,000$ generations evolving 200 candidates per generation, which took a total of 22 hours and 50 minutes. The experiment was performed on a computation server running Fedora 14 64-bit, with 8 cores running at 2 Ghz and 12 GB memory, of which max. 8 GB was used by the ETM. The four default quality dimensions of replay fitness, precision, generalization

[1] ProM is available for download from `http://www.processmining.org/`, the ETM algorithm is included in the 'EvolutionaryTreeMiner' package.

[2] More information about the CoSeLoG project can be found at `http://www.win.tue.nl/coselog/`



f: 0.701	p: 0.894	#edits
s: 0.947	g: 0.724	0

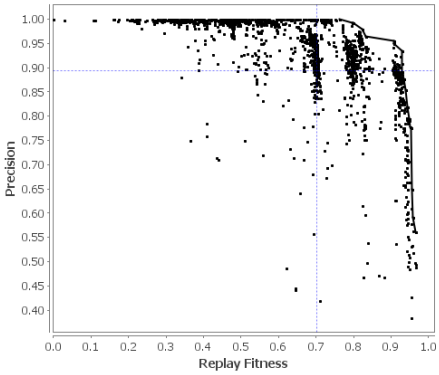
Fig. 4: Reference model for the case study

and simplicity are considered. Additionally, as a fifth dimension, the number of low-level edits made to a provided reference process model is counted as a fitness metric for the similarity dimension. This resulted in a Pareto front containing 2,562 process trees, considering 5 dimensions.

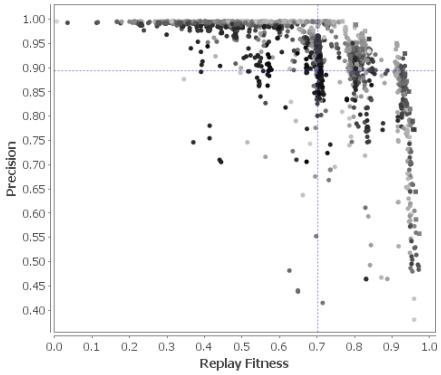
Visualizing five dimensions at once is very difficult, therefore the Pareto front can be viewed using charts where the dimensions can be plotted on the X and Y axes. Fig. 5 shows plots using the dimensions replay fitness and precision. The dot plot of Fig. 5a shows the distribution of candidates when considering the dimensions replay fitness and precision. Each dot represents a process model with that particular combination of values for the two selected dimensions. The lines on the top right connect those process models that are on the sub Pareto front, i.e. the Pareto front only considering the dimensions replay fitness and precision. All process models contained in the Pareto front are on a sub Pareto front considering one or more of the quality dimensions. This chart shows that there is not a single process model which has a very good score (i.e. 1.000) for both precision and replay fitness. Currently, the reference process model is selected in the chart, which is indicated by the dotted horizontal and vertical lines.

Additionally, a third dimension could be included in this chart by coloring the dots. This is shown in Fig. 5b where the color indicates the number of edits. A dark color means few edits and the lightest grey color indicates the maximum number of 61 edits observed. Again, the reference process model is selected in this chart. From this enhanced dot plot is becomes clear that the best process models are colored light, meaning that they require more than just a few edits. Furthermore, surrounding the selected reference process model is a 'cloud' of darker dots, indicating that only small changes in precision and replay fitness can be achieved if the process model can only be changed slightly.

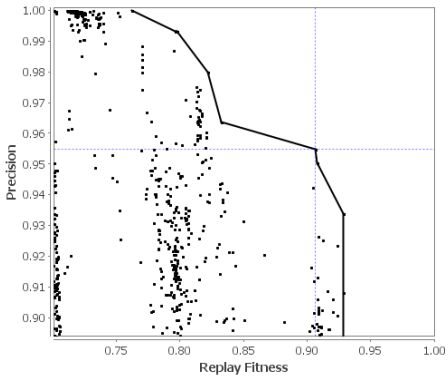
Of course, the municipality wants to know how the reference model can be improved. Fig. 5c shows a zoomed-in view of the dot plot of Fig. 5a, where only the process models with better scores for precision and replay fitness than the reference model are shown. This plot makes the trade-offs between replay fitness and precision clear. On the bottom right for instance we see the process models which have a good replay fitness, but at the cost of a bad precision score. The almost straight lines indicate that with only a very small reduction in replay



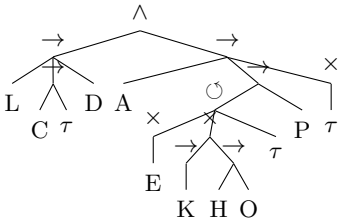
(a) Pareto front on the dimensions replay fitness and precision



(b) Pareto front on the dimensions replay fitness and precision, colored by number of edits (darker means less edits)



(c) Pareto front on the dimensions replay fitness and precision, showing only modelsfitness and precision balance with better scores than the reference model



f: 0.906	p: 0.955	# edits
s: 0.826	g: 0.744	43

(d) Process Tree with good replay fitness and precision

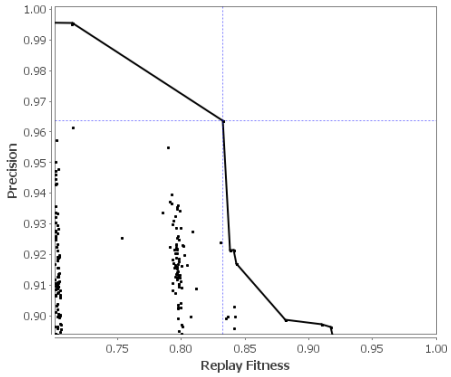
Fig. 5: Views on the Pareto front considering replay fitness and precision

fitness, the precision score can be improved significantly. This trade-off works until a precision of roughly 0.955 is achieved, then precision can only be improved by sacrificing significantly on replay fitness. Which is indicated by the almost horizontal lines between the process models.

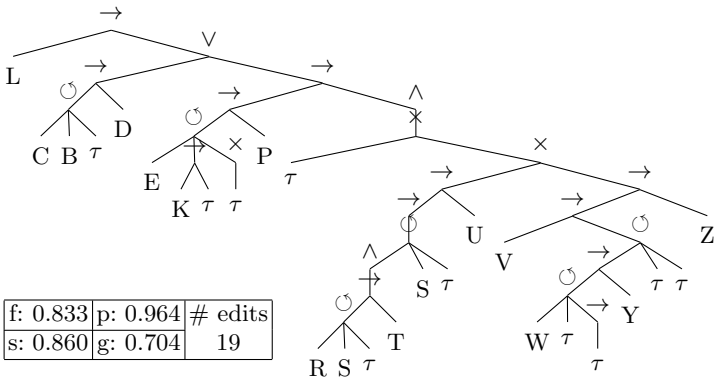
Therefore, one of the process models with the best trade-off between replay fitness and precision is the one indicated in Fig. 5c by the dotted vertical and horizontal lines. This process tree is shown in Fig. 5d. The process tree is able to replay most of the behavior observed in the five event logs, with reasonable precision. However, it required 43 edits from the reference process model. The main change is the removal of large parts of the process model, indicating that

indeed some activities are skipped often. Furthermore some frequent activities, such as activity L, are added to, or relocated within, the process model.

When looking in more detail at the Pareto front as shown in Fig. 5a it can be observed that most process models are clustered around certain values for replay fitness. For instance there is a cluster of process models around the reference process model. Then there is another cluster or models between a fitness replay score of 0.780 and 0.820, with only few process models in between. Closer investigation of the event log showed that there are six activities (*A, C, D, E, L and P*) that are executed at least $1,200$ times each. The other activities in the event log are executed at most 55 times each. Therefore these clusters consist of process models where one of the six frequent activities are in a particular good control flow construct. Since these activities are observed often, changing the control flow has a relatively large influence on the replay fitness value. The value



(a) Pareto front with maximum 20 edits



(b) Best process model with maximum 20 edits allowed

Fig. 6: Pareto front filtered to at most 20 edits from the reference process model

of replay fitness for these models can still be influenced by repositioning one of the less frequent activities, but this impact is less.

The process model shown in Fig. 5d has little resemblance to the reference process model. Therefore we filter the Pareto front to only contain process models with at most 20 edits from the reference process models. Fig. 6a shows the dot plot of the filtered Pareto front on the dimensions precision and replay fitness, using the same scales as in Fig. 5c. It is clear to see that the overall quality of the process models is worse, which can be observed by comparing the size of the area under the lines between the dots. When only 20 edits are allowed less improvements can be made. The trade-offs are also stronger, as is indicated by the almost vertical and horizontal lines between the candidates on this subfront. The process model with the best trade-off in precision and replay fitness is selected and shown in Fig. 6b. With 19 edits, a replay fitness of 0.906 and a precision of 0.955 can be obtained. This is an improvement with respect to the reference process model, especially for replay fitness. Interestingly, this process tree is also on the sub Pareto front as shown in Fig. 5c. However, when more edits are allowed, better quality models are discovered and smaller trade-offs can be made, resulting in a bigger and more detailed Pareto front.

The limitations mentioned in Sec. 2 are all resolved by using a Pareto front. The first issue of determining the weights upfront is solved by only requiring which dimensions to consider, not how to weight them against each other. As shown in Fig. 5, and the related discussion, it is possible to visualize and compare two or three dimensions, even if one is not normalized. This solves the second issue mentioned. The third issue of having only a single result is also clearly solved by presenting a Pareto front. For example between Fig. 5 and Fig. 6 it was easy to compare different trade-offs between the number of edits allowed and the resulting scores for replay fitness and precision. These insights were obtained without iteratively calling a discovery algorithm with different parameter settings. Moreover, by inspecting the Pareto front, a selection of process models can be made that satisfy a certain criteria, for instance at least as good precision and replay fitness scores as the reference process model.

5 Conclusions

In this paper we used the ETM-algorithm to construct a collection of process models that are mutually non-dominating, and thus form a Pareto front. Each of the process models in the Pareto front either scores very good in one of the considered quality dimensions or is able to balance the different quality dimensions well. We applied this extension on a real-life event log. By selecting different views on the Pareto front, we have shown that several insights can be gained and different complementary models can be inspected. This allows the user to decide which process model is best and should be used further. Moreover, the user is aware of the trade-offs to be made rather than giving full control to the discovery algorithm.

Furthermore, by applying the Pareto front, more quality dimensions and metrics can be used by the ETM-algorithm. Since there is no requirement any more to normalize and weight the values, absolute numbers can also be provided. This makes the ETM-algorithm extendable to use quality dimensions that are not directly related to process discovery. Examples are the discovery of process models that comply to a certain extend to rules, or to discover process models that minimize cost or expected risk.

Another benefit of using the Pareto front is the increase in diversity of the population. Previously the whole population would be focussed towards a specific combination of quality dimensions. However, with the diversity introduced by the Pareto front the ETM-algorithm has a large set of process models to choose from to evolve further. This is especially noticeable in the early generations of the ETM-algorithm where the Pareto front makes quick progress.

In the future we plan to further improve the visualization and navigation options for the Pareto front. It is critical that the user can navigate the collection of process models with ease and quickly gain insights. Moreover we plan to improve the selection of candidates for further evolution. This can be used to speed up the discovery of good process models. It will also help in a quick approach of the estimated Pareto front to the actual Pareto front.

References

1. W.M.P. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.
2. W.M.P. van der Aalst, A. Adriansyah, and B. F. van Dongen. Replaying History on Process Models for Conformance Checking and Performance Analysis. *WIREs Data Mining and Knowledge Discovery*, 2(2):182–192, 2012.
3. A. Awad, G. Decker, and M. Weske. Efficient Compliance Checking Using BPMN-Q and Temporal Logic. In *BPM 2008*, pages 326–341, 2008.
4. L.T. Bui, D. Essam, H.A. Abbass, and D. Green. Performance Analysis of Evolutionary Multi-Objective Optimization Methods in Noisy Environments. In *Proceedings of the 8th Asia Pacific Symposium on Intelligent and Evolutionary Systems*, pages 29–39, 2004.
5. J.C.A.M. Buijs, B.F. van Dongen, and W.M.P. van der Aalst. On the Role of Fitness, Precision, Generalization and Simplicity in Process Discovery. In *On the Move to Meaningful Internet Systems: OTM 2012*, Lecture Notes in Computer Science, pages 305–322. Springer, 2012.
6. J.C.A.M. Buijs, M. La Rosa, H.A. Reijers, B.F. van Dongen, and W.M.P. van der Aalst. Improving Business Process Models using Observed Behavior. In *Second International Symposium on Data-Driven Process Discover and Analysis, Post Proceedings*. Springer, 2013 (to appear).
7. K. Deb. Multi-Objective Optimization. In E.K. Burke and G. Kendall, editors, *Search Methodologies*, pages 273–316. Springer US, 2005.
8. K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. *Lecture notes in computer science*, 1917:849–858, 2000.

9. G. Governatori and A. Rotolo. An Algorithm for Business Process Compliance. In *JURIX*, pages 186–191, 2008.
10. T. Hiroyasu, S. Nakayama, and M. Miki. Comparison study of SPEA2+, SPEA2, and NSGA-II in Diesel Engine Emissions and Fuel Economy Problem. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 1, pages 236–242 Vol.1, 2005.
11. Vilfredo Pareto. *Cours D'Economie Politique, volume I and II.* F. Rouge, Lausanne, 1896.
12. A Pika, W.M.P. Aalst, C.J. Fidge, A.H.M. ter Hofstede, and M.T. Wynn. Predicting Deadline Transgressions Using Event Logs. In *Business Process Management 2012 Workshops*, volume 132. 2013.
13. L. Raisanen and R.M. Whitaker. Comparison and Evaluation of Multiple Objective Genetic Algorithms for the Antenna Placement Problem. *Mobile Networks and Applications*, 10(1-2):79–88, 2005.
14. E. Ramezani, D. Fahland, B.F. van Dongen, and W.M.P. van der Aalst. Diagnostic Information for Compliance Checking of Temporal Compliance Requirements. In *CAiSE Forum*. LNCS, Springer, 2013 (to appear).
15. S. Suriadi, C. Ouyang, W.M. P. van der Aalst, and A.H.M. ter Hofstede. Root Cause Analysis with Enriched Process Logs. In *Business Process Management Workshops*, pages 174–186, 2012.
16. D.A. van Veldhuizen and G.B. Lamont. Evolutionary Computation and Convergence to a Pareto Front. In *Late Breaking Papers at the Genetic Programming 1998 Conference*, pages 221–228, 1998.
17. M.T. Wynn, W.Z. Low, and W. Nauta. A Framework for Cost-Aware Process Management: Generation of Accurate and Timely Management Accounting Cost Reports. In *Asia-Pacific Conference on Conceptual Modelling*.
18. E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems. Proceedings of the EUROGEN2001 Conference, Athens, Greece, September 19-21, 2001.*