# Discovering Process Models that Support Desired Behavior and Avoid Undesired Behavior

Ali Norouzifar
RWTH Aachen University
ali.norouzifar@pads.rwth-aachen.de

Wil van der Aalst
RWTH Aachen University
wvdaalst@pads.rwth-aachen.de

## ABSTRACT

Process discovery is one of the primary process mining tasks and starting point for process improvements using event data. Existing process discovery techniques aim to find process models that best describe the observed behavior. The focus can be on recall (i.e., replay fitness) or precision. Here, we take a different perspective. We aim to discover a process model that allows for the good behavior observed, and does not allow for the bad behavior. In order to do this, we assume that we have a desirable event log ($L^+$) and an undesirable event log ($L^-$). For example, the desirable event log consists of the cases that were handled within two weeks, and the undesirable event log consists of the cases that took longer. Our discovery approach explores the tradeoff between supporting the cases in the desirable event log and avoiding the cases in the undesirable event log. The proposed framework uses a new inductive mining approach that has been implemented and tested on several real-life event logs. Experimental results show that our approach outperforms other approaches that use only the desirable event log ($L^+$). This supports the intuitive understanding that problematic cases can and should be used to improve processes.

## CCS CONCEPTS

• **Applied computing** → **Business process modeling**;

## KEYWORDS

Process mining, Process discovery, Desirable and undesirable behavior

**ACM Reference Format:**
Ali Norouzifar and Wil van der Aalst. 2023. Discovering Process Models that Support Desired Behavior and Avoid Undesired Behavior. In *The 38th ACM/SIGAPP Symposium on Applied Computing (SAC '23), March 27-March 31, 2023, Tallinn, Estonia.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3555776.3577818

## 1 INTRODUCTION

Process discovery is the problem of discovering a process model from an event log that aims to represent the real process. In many applications, we have valuable information about both desirable and undesirable behavior of processes. Some cases might be handled differently in a process due to their characteristics [4]. For example, cases with a long duration might have some undesirable characteristics that cause the delay. Similarly, the outcome of the traces, the existence of a specific activity, or a particular trace or event attribute could categorize the traces into desirable and undesirable traces. The focus of current discovery techniques is on discovering a process model that replays a single event log and avoids the non-observed behavior of the process. Ignoring the undesirable behavior may lead to process models that allow for this and generalize in an undesirable direction. In addition, this type of information helps to support the desirable behavior since the discovery algorithm can prioritize representing the specifically desirable behavior.

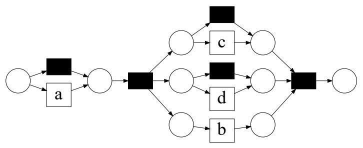
The Inductive Miner–bi (IMbi) algorithm introduced in this paper uses the problematic traces to discover better process models that are more specific to desirable traces and avoid undesirable traces. The inductive miner algorithm is one of the state-of-the-art discovery algorithms in the process mining field [8]. There are several extensions that are proposed based on this algorithm [1, 9]. It is not possible to directly adapt the inductive miner algorithm for the purpose of this paper. In each recursion, the inductive miner algorithm tries to find a cut that perfectly satisfies the cut type definitions. Considering a desirable event log and an undesirable event log, we need an algorithm to compare and rank different cuts even if they have some deviations or some important relations are missing.

## 2 MOTIVATING EXAMPLE

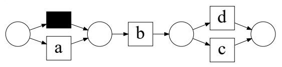
Let $\mathcal{U}_A$ be the universe of activities. A trace $\sigma = \langle a_1, a_2, \ldots, a_n \rangle \in \mathcal{U}_A^*$ is a finite sequence of activities. An event log $L \in \mathcal{B}(\mathcal{U}_A^*)$ is a multiset of traces. Consider $L^+ = [\langle a,b,c\rangle^{40}, \langle b,d\rangle^{40}, \langle c,b,d\rangle^5, \langle b,d,c\rangle^5, \langle a,d,c,b\rangle^5, \langle a,c,d,b\rangle^5]$ as a multiset of desirable traces and $L^- = [\langle b,c,d\rangle^{20}, \langle b,d,c\rangle^{20}, \langle a,c,b,d\rangle^{20}, \langle d,b,c\rangle^{20}, \langle d,c,b\rangle^{20}]$ as a multiset of undesirable traces. The goal is to discover a Petri net model that supports $L^+$ and avoids $L^-$. Figure 1a shows a process model that is discovered considering only $L^+$ using the Inductive Miner infrequent (IMf) [9] algorithm with 20% filtering. Ignoring the undesirable event log leads to this process model that perfectly fits both $L^+$ and $L^-$. Figure 1b shows a good discovered model considering both $L^+$ and $L^-$. This model avoids the behavior that is more specific to the undesirable event log and supports the behavior that is more specific to the desirable event log. In this paper, we also define novel evaluation metrics to measure such improvements.

## 3 HOW TO EVALUATE THE DISCOVERED MODELS?

Conformance checking is a type of process mining that aims to check the quality of discovered models from different perspectives. Fitness is designed to ensure the model can replay the event log



**(a) A discovered model just using $L^+$**



**(b) A discovered model considering both $L^+$ and $L^-$**

**Figure 1: A motivating example**

properly. Alignment fitness is widely accepted to measure the deviations between an event log and a process model [2]. Let $\mathcal{U}_L$ be the universe of event logs and $\mathcal{U}_M$ be the universe of Petri net models. Consider $fit_{align}{:}\mathcal{U}_L{\times}\mathcal{U}_M{\to}[0,1]$ as a function that calculates the alignment fitness. Another measurement to evaluate the fitness criterion is calculating the percentage of fitting traces. We define $fit_{trace}{:}\mathcal{U}_L{\times}\mathcal{U}_M{\to}[0,1]$ as a function that calculates this value. Precision is designed to ensure that the model does not allow for many behaviors that are not observed in the event log. $prc{:}\mathcal{U}_L{\times}\mathcal{U}_M{\to}[0,1]$ is the function that calculates escaping edges precision (ETC precision) [2]. The goal of this paper is to discover a process model that supports $L^+$ and avoids $L^-$.

The goal of this paper is to discover a process model that supports $L^+$ and avoids $L^-$. We measure this goal using the following evaluation metrics. $1{-}fit_{align}(L,M) = \overline{fit_{align}}(L,M)$ and $1{-}fit_{trace}(L,M) = \overline{fit_{trace}}(L,M)$.

- alignment accuracy: $acc_{align}{:}\mathcal{U}_L{\times}\mathcal{U}_L{\times}\mathcal{U}_M{\to}[-1,1]$ such that
$$acc_{align}(L^+,L^-,M){=}fit_{align}(L^+,M) - fit_{align}(L^-,M)$$
- trace accuracy: $acc_{trace}{:}\mathcal{U}_L{\times}\mathcal{U}_L{\times}\mathcal{U}_M{\to}[-1,1]$ such that
$$acc_{trace}(L^+,L^-,M){=}fit_{trace}(L^+,M) - fit_{trace}(L^-,M)$$
- alignment F1-score: $F1_{align}{:}\mathcal{U}_L{\times}\mathcal{U}_L{\times}\mathcal{U}_M{\to}[0,1]$ such that
$$F1_{align}(L^+,L^-,M){=}\frac{2{\times}fit_{align}(L^+,M){\times}\overline{fit_{align}}(L^-,M)}{fit_{align}(L^+,M){+}\overline{fit_{align}}(L^-,M)}$$
- trace F1-score: $F1_{trace}{:}\mathcal{U}_L{\times}\mathcal{U}_L{\times}\mathcal{U}_M{\to}[0,1]$ such that
$$F1_{trace}(L^+,L^-,M){=}\frac{2{\times}fit_{trace}(L^+,M){\times}\overline{fit_{trace}}(L^-,M)}{fit_{trace}(L^+,M)+\overline{fit_{trace}}(L^-,M)}$$

The four introduced evaluation metrics measure the same goal from different perspectives. Increasing *alignment accuracy* indicates that the discovered model can differentiate better between $L^+$ and $L^-$. It is not acceptable to discover a model with a very low $fit_{align}(L^+,M)$ but a high $acc_{align}(L^+,L^-,M)$. To avoid sacrificing $fit_{align}(L^+,M)$, we designed *alignment F1-score* to maintain a balance between increasing $fit_{align}(L^+,M)$ and increasing $\overline{fit_{align}}(L^-,M)$. *Alignment accuracy* takes only the average alignment fitness into account. Increasing *alignment accuracy* might result in discovering a process model that enforces a subprocess that is observed in neither $L^+$ nor $L^-$. This subprocess might punish $fit_{align}(L^-,M)$ more than $fit_{align}(L^+,M)$ but block the token flow path. In order to avoid such models, we propose to consider $fit_{trace}(L^+,M)$ and $fit_{trace}(L^-,M)$ to define similar evaluation metrics.

## 4 INDUCTIVE MINER–BI

Consider $L^+$ as the event log contains desirable behavior and $L^-$ as the event log contains undesirable behavior. The Inductive Miner–bi (IMbi) algorithm constructs a process model recursively. The main idea of IMbi is to consider both the desirable and the undesirable event logs in each recursion. First, it checks whether there exists a base case. If it fails to find a base case, then the algorithm tries to find an optimal cut. In Fig. 2, the general approach in each recursion is illustrated.

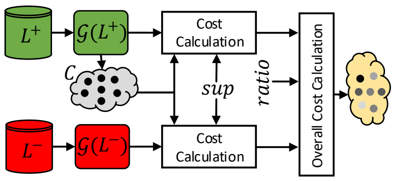


**Figure 2: Finding the optimal cut in one recursion of IMbi**

$\mathcal{G}$ is a function that extracts a directly-follows graph from an event log, such that $\mathcal{G}(L){=}(\Sigma,E)$, where

- $\Sigma{=}\{a{\in}\sigma|\sigma \in L\}$ is the set of nodes.
- $E{=}[(\sigma_i,\sigma_{i+1})|\sigma{\in}L', 1{\le}i{<}|\sigma|]$ with $L' = [\langle\triangleright\rangle.\sigma.\langle\square\rangle|\sigma \in L]$ is the multiset of edges. $\triangleright$ is the special start activity and $\square$ is the special end activity.

$\mathcal{G}(L^+)$ and $\mathcal{G}(L^-)$ in Fig. 2 represent the extracted directly-follows graphs from the desirable and undesirable event logs. The algorithm explores $\mathcal{G}(L^+)$ and constructs the set of possible cuts $C$. Each cut $c{=}(\oplus,\Sigma_1,\Sigma_2){\in}C$ divides the set of nodes into two disjoint sets, i.e., $\Sigma_1$ and $\Sigma_2$, using a cut operator $\oplus{\in}\{\to,\times,\wedge,\circlearrowleft\}$. $\to$ denotes the sequential cut, $\times$ represents the exclusive cut, $\wedge$ denotes the concurrent cut and $\circlearrowleft$ represents the loop cut.

Finding an optimal cut is not trivial. Figure 3 shows different possible cut types and their specifications. The dashed lines are optional relations. The red lines are the relations that are not allowed. The solid black lines are the mandatory relations. Based on the definitions [8], if one of the specifications of a cut type does not hold, then it should be rejected. However, this decision does not take the severity of the deviating or missing behaviors into account. In addition, we cannot compare the quality of a cut based on the desirable and undesirable event logs. To address this problem, we need to define some cost functions to quantify and compare the quality of the cuts. Considering a possible cut $c{\in}C$, $dev_G^{\oplus}(c)$ counts the number of deviating edges and $mis_G^{\oplus}(c,sup)$ assigns a cost if some edges are missing. $sup{\in}[0,1]$ is a user-defined parameter to specify how strict these cost values should be with respect to missing edges. $mis_G^{\oplus}(c,sup)$ considers more cost with higher values of $sup$ if some edges are missing. $cost_G^{\oplus}(c,sup){=}dev_G^{\oplus}(c){+}mis_G^{\oplus}(c,sup)$ assigns a cost to each cut $c{\in}C$.

$ov\_cost_{G^+,G^-}{:}C{\times}[0,1]{\times}[0,1]{\to}\mathbb{R}$ is a function that assigns an overall cost to each binary cut $c \in C$ considering both $cost_{G^+}^{\oplus}$ and $cost_{G^-}^{\oplus}$. $ratio{\in}[0,1]$ is a process discovery parameter that specifies the importance of the undesirable event log.

$$ov\_cost_{G^+,G^-}(c,sup,ratio){=}cost_{G^+}(c,sup) - ratio{\cdot}cost_{G^-}(c,sup)$$

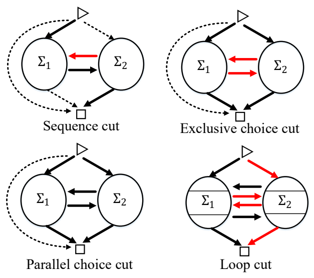


**Figure 3: Specifications of different cut types (dashed lines are optional, black lines are mandatory and red lines are deviating edges)**

The best cut in each recursion is the cut with the minimum *ov_cost*, i.e.,

$$optimal\_cut = \arg\min_{c \in C}\{ov\_cost_{G^+,G^-}(c, sup, ratio)\}$$

The algorithm continues with splitting the event logs based on the selected optimal cut and then runs the next recursion.

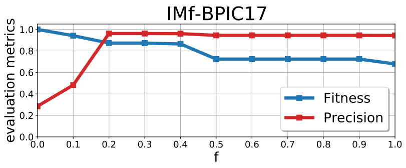
## 5 EXPERIMENT

The IMbi framework implements the approach and the source code is publicly available[1]. The BPIC 2017 event log is extracted from a loan application process. There are three types of events in this process: Application, Offer, and Workflow events. We considered the event log consisting of only Application and Offer events. The outcome of an application could be *A_pending*, *A_cancelled*, or *A_denied*. Based on the observations, if activity *“W_call incomplete files"* occurs in a trace, the chance of having *A_pending* state at the end is higher. We divided the event log into $L^+_{17}$ and $L^-_{17}$ based on the existence of activity *“W_call incomplete files"* in the traces. Among 31509 traces, 15003 traces are in $L^+_{17}$, and 16506 traces are in $L^-_{17}$. In $L^+_{17}$, 84% of the traces have *A_pending*, 6% of the traces have *A_cancelled* , and 9% of the traces have *A_denied* as the final state (few traces are incomplete). In $L^-_{17}$, 28% of the traces have *A_pending*, 57% of the traces have *A_cancelled*, and 15% of the traces have *A_denied* as the final state. The goal is to discover a process model that supports traces in $L^+_{17}$ and avoids traces in $L^-_{17}$.

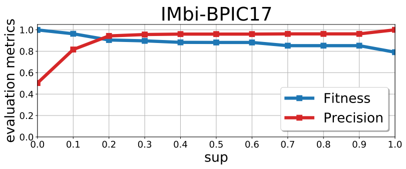
### 5.1 IMbi using Single Event Log

In this section, we show that the IMbi algorithm discovers acceptable process models from a single event log ($ratio = 0$). Otherwise, there is no guarantee that we can use this algorithm for our final goal. We use the IMf algorithm as the baseline. In this section, the infrequency filtering parameter of this algorithm is referred to as $f$. We change $f$ in the range of $[0, 1]$ and calculate the alignment fitness and the precision for different discovered models. If we set $ratio = 0$ in the IMbi algorithm, it means the undesirable event log is ignored. We can discover different process models from only the desirable event log by changing the *sup* parameter in the range of $[0, 1]$.

The alignment fitness and the precision of the discovered models using the IMf algorithm are illustrated in Fig. 4a. The IMf algorithm, after searching for the base case in each recursion, tries to find a cut that satisfies the cut type definitions. If this cut does not exist, then it filters $\mathcal{G}(L^+)$ considering $f$. The algorithm removes infrequent directly-follows relations. Then, it tries to find a cut again. Increasing $f$ in the IMf algorithm removes more infrequent directly-follows relations from $G(L^+)$. The discovered model using a high $f$ value, considers only the most frequent directly-follows relations. This leads to a decrease in the alignment fitness value and an increase in the precision value.



**(a) Discovered models from $L^+_{17}$ using IMf**



**(b) Discovered models from $L^+_{17}$ using IMbi**

**Figure 4: The fitness and precision of the discovered models from $L^+_{17}$ using IMf and IMbi (with $ratio = 0$)**

The alignment fitness and precision value of the discovered models using the IMbi algorithm are illustrated in Fig. 4b. *sup* parameter is used in the process discovery algorithm to control the strictness of the algorithm to assign some cost if a missing behavior exists. If *sup*=0, it means that the algorithm only counts the deviating edges in $\mathcal{G}(L^+)$ (red edges in Fig. 3). The discovered model may allow for more behavior than the observed behavior. The algorithm does not insist on the behavior that should be in $\mathcal{G}(L^+)$, but it is missing. By increasing the *sup* parameter, the process discovery algorithm considers some cost for the part of the behavior that is missing in $\mathcal{G}(L^+)$. This leads to discovering process models that more specifically represent the observed behaviors. Generally, as we increase the *sup* parameter, the alignment fitness value decreases, and the precision value increases. Comparing the IMf and IMbi models shows that the *sup* parameter in IMbi and $f$ parameter in IMbi have a similar effect on the fitness and precision of the discovered models. The fitness and precision value of the discovered models using the IMbi algorithm are comparable to the discovered models using the IMf algorithm.

### 5.2 IMbi using Desirable and Undesirable Event Logs

The parameter *ratio* is designed to control the involvement of the undesirable event log in the model discovery. If *ratio*=0, it means that the algorithm only considers the desirable event log, and if *ratio*=1, it means that in each recursion, the cost of a cut in $\mathcal{G}(L^+)$ and $\mathcal{G}(L^-)$ has an equal weight. Increasing *ratio* might lead to selecting the cuts that are not the best considering only $\mathcal{G}(L^+)$, but helps to support $L^+$ and avoid $L^-$. In this section, we select a *sup* parameter value based on the harmonic mean of the alignment fitness

[1]https://github.com/aliNorouzifar/InductiveMiner_bi.git

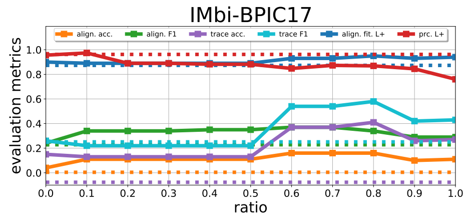


**Figure 5: Evaluation metrics of the discovered models using IMbi with $sup$ = 0.3 (dashed lines are calculated for the baseline)**

and precision. Then, we change the $ratio$ parameter in the range of $[0, 1]$ to show the effect on the evaluation metrics. In addition to the introduced evaluation metrics, we show the alignment fitness and precision corresponding to only the desirable event log.

For BPIC 2017, we select $sup$=0.3, since the harmonic mean of fitness and precision is the best based on Fig. 4b. In Fig. 5, the calculated evaluation metrics are depicted. In this figure, the dashed lines are calculated for the model discovered from $L^+_{17}$ using the IMf algorithm with $f$=0.3 as the baseline. The alignment accuracy of the baseline model is around 0, and the trace accuracy is -0.07. The discovered model cannot distinguish well between the desirable event log and the undesirable event log. The evaluation metrics for the discovered model using the IMbi algorithm with $ratio = 0$ and $sup = 0.3$ is slightly better since it has a better trace accuracy, but still has difficulties differentiating between the desirable and undesirable event logs. Increasing the $ratio$ parameter shows that the evaluation metrics improve when we consider the undesirable event log in the process discovery. The main improvement in the evaluation metrics is achieved after setting the $ratio$ parameter to 0.6. At $ratio$=0.6, all evaluation metrics except precision increase. After $ratio$=0.8, the evaluation metrics decrease since the selected cuts in the recursions are influenced by the undesirable event log, and the low precision leads to the allowance for some undesirable behaviors in the discovered models.

## 6 RELATED WORK

The focus of this paper is on discovering a process model that allows for the desirable behavior and avoids the undesirable behavior of a given process. This fresh look is different from the existing works in the literature. In [7] and [3], the authors use artificial negative events to improve the process discovery. The goal is to maintain a balance between the allowed behavior by the model and the observed behavior in the event log. Improving the precision, generalization, and complexity of the process models are the outcomes of their research. In [10], the discovery phase is considered as a binary classification and the information about undesired behavior is used to improve the discovered models. Machine learning evaluation metrics are used to evaluate these models. The focus of [5] and [10] is on discovering models to classify the traces into desirable and undesirable. In our paper, we formalized our approach as a method to discover Petri net models that represent the process and the designed evaluation metrics reflect the descriptive aspects of the discovered process models. In [6], a methodology is proposed to repair discovered models to allow for a part of the behavior that has a positive impact on process KPIs.

## 7 CONCLUSION

The experiments show that the discovered models using our approach outperform other approaches that use only desirable event logs. Based on the designed evaluation metrics, changing $sup$ and $ratio$ parameters lead to good candidate models. For some parameter configurations, the evaluation metrics could have some conflicts. Selecting the best process model depends on the application. For example, if we improve the alignment accuracy, it may have an inverse effect on the trace accuracy. The discovery algorithm may choose some cuts that are not good enough based on neither $L^+$ nor $L^-$, but punishing $L^-$ behavior may lead to this decision. Adding some process knowledge to the discovery algorithm helps to avoid bad decisions. For example, a set of rules extracted from process knowledge can be used to guide the algorithm. IMbi has a high computational cost due to the many possible candidate cuts that needs to be improved in the next steps. Some differences between $L^+$ and $L^-$ might not be in the process structures. Our algorithm only takes the control flow into account and discovers a Petri net that has a different structure in order to support the desirable event log and avoid the undesirable event log.

## ACKNOWLEDGMENTS

This research was supported by the research training group "Dataninja" (Trustworthy AI for Seamless Problem Solving: Next Generation Intelligence Joins Robust Data Analysis) funded by the German federal state of North Rhine-Westphalia.

## REFERENCES

[1] Dennis Brons, Roeland Scheepens, and Dirk Fahland. 2021. Striking a new balance in accuracy and simplicity with the probabilistic inductive miner. In *2021 3rd International Conference on Process Mining (ICPM)*. IEEE, 32–39.
[2] Josep Carmona, Boudewijn F. van Dongen, Andreas Solti, and Matthias Weidlich. 2018. *Conformance Checking - Relating Processes and Models*. Springer.
[3] Hernán Ponce De León, Lucio Nardelli, Josep Carmona, and Seppe KLM vanden Broucke. 2018. Incorporating negative information to process discovery of complex systems. *Information Sciences* 422 (2018), 480–496.
[4] Massimiliano De Leoni, Wil M. P. van der Aalst, and Marcus Dees. 2016. A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs. *Information Systems* 56 (2016), 235–257.
[5] Hugo De Oliveira, Vincent Augusto, Baptiste Jouaneton, Ludovic Lamarsalle, Martin Prodel, and Xiaolan Xie. 2020. An optimization-based process mining approach for explainable classification of timed event logs. In *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. IEEE, 43–48.
[6] Marcus Dees, Massimiliano de Leoni, and Felix Mannhardt. 2017. Enhancing process models to improve business performance: A methodology and case studies. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer, 232–251.
[7] Stijn Goedertier, David Martens, Jan Vanthienen, and Bat Baesens. 2009. Robust process discovery with artificial negative events. *Journal of Machine Learning Research* 10 (2009), 1305–1340.
[8] Sander JJ Leemans, Dirk Fahland, and Wil M. P. van der Aalst. 2013. Discovering block-structured process models from event logs-a constructive approach. In *International conference on applications and theory of Petri nets and concurrency*. Springer, 311–329.
[9] Sander JJ Leemans, Dirk Fahland, and Wil M. P. van der Aalst. 2013. Discovering block-structured process models from event logs containing infrequent behaviour. In *International conference on business process management*. Springer, 66–78.
[10] Tijs Slaats, Søren Debois, and Christoffer Olling Back. 2021. Weighing the Pros and Cons: Process Discovery with Negative Examples. In *International Conference on Business Process Management*. Springer, 47–64.