ILP-Based Process Discovery Using Hybrid Regions

S.J. van Zelst, B.F. van Dongen, and W.M.P. van der Aalst

Department of Mathematics and Computer Science
Eindhoven University of Technology, The Netherlands
{s.j.v.zelst,b.f.v.dongen,w.m.p.v.d.aalst}@tue.nl

Abstract. The *language-based theory of regions*, stemming from the area of *Petri net synthesis*, forms a fundamental basis for Integer Linear Programming (ILP)-based process discovery. Based on example behavior in an event log, a process model is derived that aims to describe the observed behavior. Building on top of the existing ILP-formulation, we present a new ILP-based process discovery formulation that unifies two existing types of language-based regions and, additionally, we present a generalized ILP objective function that captures both region-types and helps us to find suitable process discovery results.

Keywords: Process mining, process discovery, integer linear programming, region theory

1 Introduction

Process Mining [1] forms the connecting element between business process modeling and data analysis. Its main aim is to extract knowledge from business process execution data originating from a variety of data sources, e.g., enterprise information systems, social media, embedded systems etc. Within process mining, three main branches are distinguished being *process discovery*, *conformance checking* and *process enhancement*. Within process discovery the aim is to, given an event log, *reconstruct* a process model. Within conformance checking the aim is to assess, given a process model and an event log, the *conformance* of the event log to the process model. Within process enhancement the aim is to further improve and/or align existing process models by combining the two aforementioned disciplines, e.g., identification and removal of bottlenecks within business processes.

The area of *Petri net synthesis* [2] is concerned with deciding whether there exists a Petri net that *exactly* describes a given *sequential behavioral system description*. A subclass of synthesis techniques is the area of *region theory* which is both defined on transition systems, known as *state-based region theory*, and on languages, known as *language-based region theory*.

Language-based region theory forms a fundamental basis for ILP-based process discovery [3]. Within process discovery however, a process model is typically valued w.r.t. four essential process mining quality dimensions being *replay-fitness*, *precision*, *generalization* and *simplicity* [4]. Applying traditional Petri net synthesis techniques in a process discovery context will result in models that have perfect replay-fitness, maximal precision, low generalization and often low simplicity.

In this paper we propose a unified approach w.r.t. ILP-based process discovery, based on the newly introduced concept of *hybrid variable-based regions*. Hybrid variable-based regions unite two existing language-based region definitions and allow us to vary the number of variables used within the ILP problems being solved. Therefore, we assess the impact of using a different number of variables on the average computation time of solving ILPs during ILP-based process discovery. Additionally we present a new generalized ILP objective function that supports hybrid variable-based regions and show that the objective function yields correct process discovery results.

The outline of this paper is as follows. In Section 2 we present basic preliminaries. In Section 3 we present language-based regions. In Section 4 we show how to adopt language-based regions within process discovery using ILP. In Section 5 we present a brief evaluation of the performance of the approach. Section 6 covers related work. Section 7 concludes the paper.

2 Preliminaries

2.1 Bags, Sequences and Vectors

Let $\mathbb{Z}$ denote the set of integers, let $\mathbb{N}$ denote the set of positive integers *including* 0 and let $\mathbb{N}^+$ denote the set of positive integers *excluding* 0. Let $\mathbb{R}$ denote the set of real numbers and let $\mathbb{R}^+$ denote the set of positive real numbers excluding 0.

Let S denote a set. Let us define a *bag* B as a function $B : S \to \mathbb{N}$. Notation-wise we write a bag as $[e_1^{n_1}, e_2^{n_2}, ..., e_3^{n_3}]$, where $e_i \in S$, $n_i \in \mathbb{N}^+$ and $e_i^{n_i} \equiv B(e_i) = n_i$. If for some element e, $B(e) = 1$, we omit its superscript. An empty bag is denoted as $\emptyset$. As an example let $S_1 = \{a, b, c, d\}$ and let B_1 denote a bag consisting of 3 a's, 5 b's, 1 c and 0 d's, i.e. $B_1 = [a^3, b^5, c]$. Element inclusion applies to bags as well, i.e. given $e \in S$ and $B(e) > 0$ then also $e \in B$. Thus we have $a \in B_1$ whereas $d \notin B_1$.

A *sequence* σ is a function relating positions to elements $e \in S$, i.e. $\sigma : \{1, 2, ..., k\} \to S$. An empty sequence is denoted as ϵ. We write every non-empty sequence as $\langle e_1, e_2, ..., e_k \rangle$. The set of all possible sequences over a set S is denoted as S^*. We define *concatenation* of sequences σ_1 and σ_2 as $\sigma_1 \cdot \sigma_2$, e.g., $\langle a, b \rangle \cdot \langle c, d \rangle = \langle a, b, c, d \rangle$. If we are given a set X of sequences over S, i.e., $X \subseteq S^*$, we define X's *prefix-closure* as $\overline{X} = \{\sigma_1 \in S^* | \exists_{\sigma_2 \in S^*} (\sigma_1 \cdot \sigma_2 \in X)\}$. Let $X \subseteq S^*$, X is *prefix-closed* if $X = \overline{X}$. Let $X \subseteq S^*$ and let $B_X : X \to \mathbb{N}$ be a bag, we define B_X's prefix closure as $\overline{B}_X : \overline{X} \to \mathbb{N}$

$$\overline{B}_X(\sigma) = B(\sigma) + \sum_{\sigma \cdot \langle e \rangle \in \overline{X}} \overline{B}_X(\sigma \cdot \langle e \rangle)$$

Thus, for set $X_1 = \{\langle a, b \rangle, \langle a, c \rangle\}$ we have $\overline{X}_1 = \{\epsilon, \langle a \rangle, \langle a, b \rangle, \langle a, c \rangle\}$ whereas for bag $B_2 = [\langle a, b \rangle^5, \langle a, c \rangle^3]$ we have $\overline{B}_2 = [\epsilon^8, \langle a \rangle^8, \langle a, b \rangle^5, \langle a, c \rangle^3]$.

Let S be a set on which we can pose some total order and let $R \subseteq \mathbb{R}$ be a range of values. Vectors are denoted as $\vec{z} \in R^{|S|}$, where $\vec{z}(e) \in R$ and $e \in S$. A vector $\vec{z}$ represents a column vector. For vector multiplication we assume vectors to agree on their indices. Thus, given a totally ordered set $S = \{e_1, e_2, ..., e_n\}$ and $\vec{z}_1, \vec{z}_2 \in R^{|S|}$ we have $\vec{z}_1^\intercal \vec{z}_2 = \sum_{i \in \{1,2,...,n\}} \vec{z}_1(e_i)\vec{z}_2(e_i)$. *Parikh vectors* represent the number of occurrences of a certain element within a sequence. A Parikh vector is a function

$\vec{p} : S^* \rightarrow \mathbb{N}^{|S|}$ with $\vec{p}(\sigma) = (\sigma_{\uparrow e_1}, \sigma_{\uparrow e_2}, ..., \sigma_{\uparrow e_n})$ where $\sigma_{\uparrow e_i} = |\{j \mid 1 \leq j \leq |\sigma|, \sigma(j) = e_i\}|$. As an example consider $S = \{a, b, c, d\}$, $\sigma_1 = \langle a, b, d \rangle$ and $\sigma_2 = \langle a, c, d \rangle$. We have $\vec{p}(\sigma_1)^\intercal = (1, 1, 0, 1)$ and $\vec{p}(\sigma_2)^\intercal = (1, 0, 1, 1)$. Note that $\sigma_{1\uparrow b} = 1$ whereas $\sigma_{2\uparrow b} = 0$. Given a Parikh vector $\vec{p} : S^* \rightarrow \mathbb{N}^{|S|}$ and a set $Q \subseteq S$, we define $\vec{p}_Q : S^* \rightarrow \mathbb{N}^{|Q|}$. Thus given $S = \{a, b, c, d\}$ and $\sigma_1 = \langle a, b, d \rangle$ we have $\vec{p}_{\{a,c,d\}}(\sigma_1)^\intercal = (1, 0, 1)$.

2.2 Event Logs and Petri Nets

The main goal of process discovery is to describe the observed behavior within an event log by means of a process model. Thus, event logs act as the input of process discovery whereas some process model acts as the output of process discovery. An abstract example of an event log is presented in Table 1. Often more data attributes are available in an event log, e.g., *customer id*, *credit balance* etc. In this paper we focus on the sequential ordering of *activities* w.r.t. *cases*, i.e. the *control-flow perspective*.

Definition 1 (Event log). *Let A be a set of activities, an event log L is a bag of sequences over A, i.e., $L : A^* \rightarrow \mathbb{N}$.*[1]

A sequence of activities $\sigma \in L$ is referred to as a *trace*. We assume that any given set of activities A is totally ordered (or we are able to trivially pose a total ordering on A).

We consider Petri nets to describe process models. A Petri net is a bipartite graph consisting of a set of vertices called *places* and a set of vertices called *transitions*. Arcs (directed edges) are connecting places and transitions and have an associated positive weight.

Definition 2 (Petri net). *A Petri net is a 3-tuple $N = (P, T, W)$, where P is a set of places and T is a set of transitions with $P \cap T = \emptyset$. W is the weighted flow relation of N, $W : (P \times T) \cup (T \times P) \rightarrow \mathbb{N}$.*

For a given node $x \in P \cup T$, the pre-set of x in N is defined as $\bullet x = \{y \mid W(y, x) > 0\}$. Correspondingly $x\bullet = \{y \mid W(x, y) > 0\}$ denotes the post-set of x in N. Graphically we represent places as *circles* and transitions as *boxes*. For every $(x, y) \in (P \times T) \cup (T \times P)$ with $W(x, y) > 0$ we draw an *arc* from x to y. If $W(x, y) > 1$

Table 1: An abstract example of an *event log*.

Case id	Activity id	Resource id	Time-stamp
c1	*a*	*Lucy*	*2015-01-05T09:13:37+00:00*
c2	*a*	*John*	*2015-01-05T13:37:25+00:00*
c2	*b*	*Pete*	*2015-01-06T13:14:15+00:00*
c2	*d*	*Lucy*	*2015-01-06T15:27:18+00:00*
c1	*c*	*Pete*	*2015-01-07T14:28:56+00:00*
c1	*d*	*John*	*2015-01-07T15:30:00+00:00*
⋮	⋮	⋮	⋮

[1] In practice we extract an event log L from some information system. Consequently A is implicitly defined by the event log, i.e., A only consists of events that are actually present L.

we denote the arc's weight $W(x, y)$ on top of the arc from x to y. A Petri net is *pure* if it does not contain self-loops, i.e., if $W(x, y) > 0$ then $W(y, x) = 0$. An example (pure) Petri net is depicted in Figure 1.

Definition 3 (Marked Petri net). *Let $N = (P, T, W)$ be a Petri net. A marking of N is a bag over N's places, i.e. $M : P \to \mathbb{N}$. A marked Petri net is a pair (N, M_0), where M_0 represents N's initial marking.*

Let $N = (P, T, W)$ be a Petri net and let M be a marking of N. A transition $t \in T$ is *enabled*, denoted $(N, M)[t\rangle$, if and only if $\forall_{p \in {}^\bullet t}(M(p) \geq W(p, t))$. Graphically we represent a marking by drawing exactly a place's marking-multiplicity number of dots inside the place (e.g. p_1 in Figure 1 with $M_0 = [p_1]$). If a transition t is enabled in (N, M), t may *fire*, resulting in a new marking M'. When t fires, denoted as $(N, M)[t\rangle(N, M')$, we have $\forall_{p \in P}(M'(p) = M(p) - W(p, t) + W(t, p))$. For example in Figure 1 we have $(N_1, [p_1])[a\rangle(N_1, [p_2])$.

Definition 4 (Firing sequences). *Let $N = (P, T, W)$ be a Petri net and let (N, M_0) be a corresponding marked Petri net. Sequence $\sigma = \langle t_1, t_2, ..., t_n\rangle \in T^*$ is a firing sequence of (N, M_0), written as $(N, M_0)[\sigma\rangle(N, M_n)$ if and only if for $n = |\sigma|$ there exist markings $M_0, M_1, M_2, ..., M_n$ such that $(N, M_0)[t_1\rangle(N, M_1)$, $(N, M_1)[t_2\rangle(N, M_2), ..., (N, M_{n-1})[t_n\rangle(N, M_n)$.*

Note that infinitely many firing sequences can exist given a Petri net. Some example firing sequences of the Petri net depicted in Figure 1 are: $(N_1, [p_1])[\langle a\rangle\rangle(N_1, [p_2])$, $(N_1, [p_1])[\langle a, b\rangle\rangle(N_1, [p_3])$ and $(N_1, [p_1])[\langle a, c, d, e, f, e, f, e, g\rangle\rangle(N_1, \emptyset)$. The set of all possible firing sequences in a Petri net N is called $N's$ language, i.e., all sequences $\sigma \in T^*$ s.t. $(N, M_0)[\sigma\rangle(N, M_i)$. N's language is denoted as $\mathcal{L}(N) \subseteq T^*$ and is prefix-closed.

Consider Figure 1 and event log $L_1 = [\langle a, b, d, e, g\rangle^{10}, \langle a, c, d, e, g\rangle^{9}, \langle a, b, d, e, f, e, g\rangle^{11}, \langle a, c, d, e, f, e, g\rangle^{8}]$ over $A_1 = \{a, b, c, d, e, f, g\}$. Clearly we have $\overline{L}_1 \subset \mathcal{L}(N_1)$, and thus replay-fitness of L_1 on N_1 is perfect. We will use N_1, A_1 and L_1 throughout the paper as a running-example.

3 Regions

The concept of regions forms the basis of region theory within Petri net synthesis. Given an event log L over a set of activities A, language-based regions are used to represent

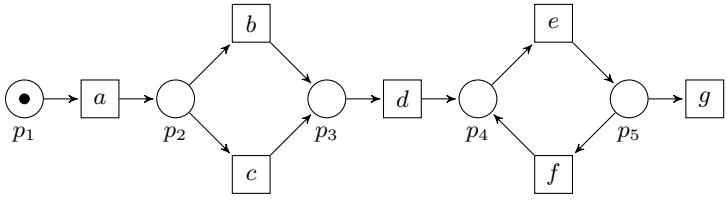

Fig. 1: A pure Petri net $N_1 = (P_1, T_1, W_1)$ with $P_1 = \{p_1, p_2, ..., p_5\}, T_1 = \{a, b, ..., g\}$ and $W_1(p_1, a) = W_1(a, p_2) = ... = W_1(p_5, g) = 1$.

places in a resulting Petri net $N = (P, A, W)$. A language-based region maps every $a \in A$ to an integer representing arc-weight. For such mapping over A to be a region we pose the restriction that the corresponding place $p \in P$ should not block any $\sigma \in \overline{L}$, i.e., $\sigma \in \overline{L} \Rightarrow \sigma \in \mathcal{L}(N)$. We identify two basic definitions of language-based regions which we classify as *single variable-based regions* and *dual variable-based regions*. The main difference is the number of variables used to define a region.

Definition 5 (Single variable-based regions). *Let L be an event log over a set of activities A. Let $m \in \mathbb{N}$ and let $\vec{v} \in \mathbb{Z}^{|A|}$. A pair $r = (m, \vec{v})$ is a single variable-based region iff:*

$$\forall_{\sigma \in \overline{L}} (m + \vec{p}(\sigma)^{\intercal} \vec{v} \geq 0) \tag{3.1}$$

Let $R^s(\overline{L})$ denote the set of all possible single variable-based regions of $\overline{L}$. Equation 3.1 generates a set of linear inequalities, i.e. applying Equation 3.1 on L_1 yields:

$$\begin{array}{rll}
m & \geq 0 & \epsilon \\
m + \vec{v}(a) & \geq 0 & \langle a \rangle \\
m + \vec{v}(a) + \vec{v}(b) & \geq 0 & \langle a, b \rangle \\
\vdots & \vdots & \vdots \\
m + \vec{v}(a) + \vec{v}(b) + \vec{v}(d) + 2\vec{v}(e) + \vec{v}(f) + \vec{v}(g) & \geq 0 & \langle a, b, d, e, f, e, g \rangle \\
m + \vec{v}(a) + \vec{v}(c) + \vec{v}(d) + 2\vec{v}(e) + \vec{v}(f) + \vec{v}(g) & \geq 0 & \langle a, c, d, e, f, e, g \rangle
\end{array}$$

Single variable-based regions use one *single* decision variable for each $a \in A$, represented by $\vec{v} \in \mathbb{Z}^{|A|}$. Expressing a single variable-based region $r = (m, \vec{v})$ as a place $p \in P$ in a marked net (N, M_0) with $N = (P, A, W)$ is straightforward. We have $M_0(p) = m$, if $\vec{v}(a) > 0$ then $W(a, p) = \vec{v}(a)$, if $\vec{v}(a) < 0$ then $W(p, a) = -\vec{v}(a)$ and if $\vec{v}(a) = 0$ then $W(a, p) = W(p, a) = 0$. Consider place p_2 depicted in Figure 1 which can be represented as a single variable-based region $r = (m, (\vec{v}(a), \vec{v}(b), ..., \vec{v}(g))) = (0, (1, -1, -1, 0, 0, 0, 0))$. Note that for each inequality generated by Equation 3.1, place p_2 has a value of at least 0 and hence is a member of $R^s(\overline{L})$.

Single variable-based regions only allow us to synthesize/discover pure Petri nets. As a consequence we can not find self-loops, i.e. we can not find a place p in a resulting net $N = (P, A, W)$ s.t. $p \in {\bullet a} \cap a^{\bullet}$, for any $a \in A$. A lot of workflow patterns [5] in fact exhibit places that include self-loops, e.g. milestone patterns, mutual-exclusion patterns, priority patterns etc. Hence, we define *dual variable-based regions* which explicitly allow us to distinguish between incoming and outgoing arcs.

Definition 6 (Dual variable-based regions). *Let L be an event log over a set of activities A. Let $m \in \mathbb{N}$ and $\vec{x}, \vec{y} \in \mathbb{N}^{|A|}$. A triple $r = (m, \vec{x}, \vec{y})$ is a dual variable-based region iff:*

$$\forall_{\sigma = \sigma' \cdot \langle a \rangle \in \overline{L}} (m + \vec{p}(\sigma')^{\intercal} \vec{x} - \vec{p}(\sigma)^{\intercal} \vec{y} \geq 0) \tag{3.2}$$

Let $R^d(\overline{L})$ denote the set of all possible dual variable-based regions of $\overline{L}$. Like Definition 5, Definition 6 generates a set of linear inequalities. Applying Definition 6 on L_1 yields:

$$\begin{array}{rll}
m - \vec{y}(a) & \geq 0 & \langle a \rangle \\
m - \vec{y}(a) + \vec{x}(a) - \vec{y}(b) & \geq 0 & \langle a, b \rangle \\
m - \vec{y}(a) + \vec{x}(a) - \vec{y}(c) & \geq 0 & \langle a, c \rangle \\
\vdots & \vdots & \vdots \\
m - \vec{y}(a) + \vec{x}(a) - \vec{y}(b) + \vec{x}(b) - \vec{y}(d) + \vec{x}(d) - 2\vec{y}(e) + 2\vec{x}(e) - \vec{y}(f) + \vec{x}(f) - \vec{y}(g) & \geq 0 & \langle a, b, d, e, f, e, g \rangle \\
m - \vec{y}(a) + \vec{x}(a) - \vec{y}(c) + \vec{x}(c) - \vec{y}(d) + \vec{x}(d) - 2\vec{y}(e) + 2\vec{x}(e) - \vec{y}(f) + \vec{x}(f) - \vec{y}(g) & \geq 0 & \langle a, c, d, e, f, e, g \rangle
\end{array}$$

Dual variable-based regions use two decision variables per $a \in A$, represented by $\vec{x}, \vec{y} \in \mathbb{N}^{|A|}$. The variables allow us to distinguish between *incoming arcs* and *outgoing arcs* when translating regions to Petri nets. Expressing a dual variable-based region $r = (m, \vec{x}, \vec{y})$ as a place $p \in P$ in marked net (N, M_0) with $N = (P, A, W)$ is again straightforward. We have $M_0(p) = m$, $W(a, p) = \vec{x}(a)$ and $W(p, a) = \vec{y}(a)$. Again consider place p_2 depicted in Figure 1 which can be represented as a dual variable-based region $r = (m, (\vec{x}(a), \vec{x}(b), ..., \vec{x}(g)), (\vec{y}(a), \vec{y}(b), ..., \vec{y}(g))) = (0, (1, 0, 0, 0, 0, 0, 0), (0, 1, 1, 0, 0, 0, 0))$. Verify that for each linear in-equality generated by Definition 6, place p_2 has a value of at least 0 and hence is a member of $R^d(\overline{L})$.

4 Hybrid Variable-Based Regions in Process Discovery

Using dual variable-based regions allows us to express non-pure places, i.e, self-loops, milestones etc. However, this type of regions uses roughly twice as many variables compared with single variable-based regions. To balance the number of variables used, though still enhance the possibility of finding non-pure Petri net structures, we introduce the new concept of hybrid regions, capturing both single and dual variable-based regions.

Definition 7 (Hybrid variable-based regions). *Let L be an event log over a set of activities A. Let $A_s, A_d \subseteq A$ be two sets of activities with $A_s \cup A_d = A$ and $A_s \cap A_d = \emptyset$. Let $m \in \mathbb{N}$, $\vec{v} \in \mathbb{Z}^{|A_s|}$ and $\vec{x}, \vec{y} \in \mathbb{N}^{|A_d|}$. A quadruple $r = (m, \vec{v}, \vec{x}, \vec{y})$ is a hybrid variable-based region iff:*

$$\forall_{\sigma=\sigma' \cdot \langle a \rangle \in \overline{L}} (m + \vec{p}_{A_s}(\sigma)^\intercal \vec{v} + \vec{p}_{A_d}(\sigma')^\intercal \vec{x} - \vec{p}_{A_d}(\sigma)^\intercal \vec{y} \geq 0) \tag{4.1}$$

Given a set of activities A and two sets of activities $A_s, A_d \subseteq A$ with $A_s \cup A_d = A$ and $A_s \cap A_d = \emptyset$, we refer to a *hybrid partition* of A. If we choose $A_d = A$, Equation 4.1 is equal to Equation 3.2. If we choose $A_s = A$, Equation 4.1 can be reformulated as:

$$\forall_{\sigma \in \overline{L} \setminus \{\epsilon\}} (m + \vec{p}(\sigma)^\intercal \vec{v} \geq 0) \tag{4.2}$$

Equation 4.2 does not equal Equation 3.1, however, as $\vec{p}(\epsilon) = \vec{0}$ and $m \in \mathbb{N}$, the equations are equivalent in this context.

Note that the set of hybrid variable-based regions, i.e. the set of variable assignments that adhere to Definition 7 depends on the hybrid partition of A into A_s and A_d. Therefore, we let A_s act as a parameter for the set of feasible hybrid variable-based regions. Let $R^h_{A_s}(\overline{L})$ denote the set of all possible hybrid variable-based regions of $\overline{L}$ where A_s represent a hybrid partition of A. Note $R^h_A(\overline{L}) = R^s(\overline{L})$ and $R^h_\emptyset(\overline{L}) = R^d(\overline{L})$. Region $r = (0, \vec{0}, \vec{0}, \vec{0})$ is deemed the *trivial region*.

All three language-based region definitions, i.e. single, dual and hybrid variable-based regions, provide means to accept and/or reject potential places in a to be constructed Petri net. In classical Petri net synthesis approaches, one keeps looking for feasible places until either $\overline{L} = \mathcal{L}(N)$, or if this is impossible, $\mathcal{L}(N) \setminus \overline{L}$ is minimized. Unfortunately, most models returned by classical Petri net synthesis techniques result in models that are unusable from a process discovery perspective. Hence, we need to relax the strict formal requirements posed on the relation between $\overline{L}$ and $\mathcal{L}(N)$.

Definition 8 (Hybrid variable-based process discovery ILP-formulation). *Let L be an event log over a set of activities A and let $A_s, A_d \subseteq A$ be a hybrid partition. Let $\mathbf{M}_s, \mathbf{M}_d, \mathbf{M}'_d$ be three matrices where $\mathbf{M}_s$ is an $|\overline{L}| \setminus \{\epsilon\} \times A_s$ matrix with $\mathbf{M}_s(\sigma, a) = \vec{p}(\sigma)(a)$ and $\mathbf{M}_d$, $\mathbf{M}'_d$ are two $|\overline{L}| \setminus \{\epsilon\} \times A_d$ matrices with $\mathbf{M}_d(\sigma, a) = \vec{p}(\sigma)(a)$ and $\mathbf{M}'_d(\sigma, a) = \vec{p}(\sigma')(a)$ (where $\sigma = \sigma' \cdot \langle a' \rangle \in \overline{L}$). Let $c_m \in \mathbb{R}$, $\vec{c_v} \in \mathbb{R}^{|A_s|}$ and $\vec{c_x}, \vec{c_y} \in \mathbb{R}^{|A_d|}$. The hybrid variable-based process discovery ILP-formulation, denoted $ILP^h_{\overline{L}}$, is defined as:*

$$
\begin{array}{lrl}
\textbf{\textit{minimize}} & z = c_m m + \vec{c_v}^{\intercal}\vec{v} + \vec{c_x}^{\intercal}\vec{x} + \vec{c_y}^{\intercal}\vec{y} & \textit{objective function} \\
\textbf{\textit{such that}} & m\vec{1} + \mathbf{M}_s\vec{v} + \mathbf{M}'_d\vec{x} - \mathbf{M}_d\vec{y} \geq \vec{0} & \textit{hybrid variable-based region} \\
\textbf{\textit{and}} & -\vec{1} \leq \vec{v} \leq \vec{1} & \textit{i.e. } \vec{v} \in \{-1, 0, 1\}^{|A_s|} \\
& \vec{0} \leq \vec{x} \leq \vec{1} & \textit{i.e. } \vec{x} \in \{0, 1\}^{|A_d|} \\
& \vec{0} \leq \vec{y} \leq \vec{1} & \textit{i.e. } \vec{y} \in \{0, 1\}^{|A_d|} \\
& 0 \leq m \leq 1 & \textit{i.e. } m \in \{0, 1\}
\end{array}
$$

The ILP-fromulation presented in Definition 8 uses the set of linear in-equalities generated by Equation 4.1 within its constraint body. The formulation however binds $\vec{v}$ to $\{-1, 0, 1\}^{|A_s|}$ and $\vec{x}, \vec{y}$ to $\{0, 1\}^{|A_d|}$, i.e. the formulation only allows for discovering Petri nets with arc weights restricted to $\{0, 1\}$. Additionally it defines an objective function, i.e. $z = c_m m + \vec{c_v}^{\intercal}\vec{v} + \vec{c_x}^{\intercal}\vec{x} + \vec{c_y}^{\intercal}\vec{y}$, that maps each region to a real value, i.e. $z : R^h_{A_s}(\overline{L}) \rightarrow \mathbb{R}$. In general we are free to choose whatever objective function we like, however, using different objective functions will lead to different process discovery results. Choosing $c_m = 0$ and $\vec{c_v} = \vec{c_x} = \vec{c_y} = \vec{1}$ leads to arc minimization whereas maximizing the same objective leads to arc maximization, resulting in different places/regions found by the underlying ILP solver. The objective function proposed in [3], being a minimization function, tries to minimize the number of incoming arcs and maximize the number of outgoing arcs. The objective function can be defined in terms of $c_m, \vec{c_v}, \vec{c_x}$ and $\vec{c_y}$ as $c_m = |\overline{L}|$, $\vec{c_v} = \sum_{\sigma \in \overline{L}} \vec{p}_{A_s}(\sigma)$, $\vec{c_x} = \sum_{\sigma \in \overline{L}} \vec{p}_{A_d}(\sigma)$ and $\vec{c_y} = -\vec{c_x}$, i.e.:

$$
z(r) = \sum_{\sigma \in \overline{L}} (m + \vec{p}_{A_s}(\sigma)^{\intercal}\vec{v} + \vec{p}_{A_d}(\sigma)^{\intercal}(\vec{x} - \vec{y})) \tag{4.3}
$$

4.1 Optimizing Token Throughput

The objective function used within [3], i.e. Equation 4.3 is less generic as the one proposed in Definition 8. As shown in [3], it favors *minimal* regions. A minimal region is a region that is not expressible as the sum of two other regions. The objective function is solely based on the set-representation of the prefix-closure of an event log. However, an event log is a bag of traces and thus consists of information on trace frequency. We propose a generalized *prefix-closure-based* objective function that incorporates a word-based scaling function β. The scaling function β is required to map all sequences in the prefix-closure of the event log to some positive real value. The actual implementation is up to the user, although we present an instantiation of β that works well for process discovery. We show that the proposed generalized weighted prefix-closure-based hybrid region objective function favors minimal regions, given any scaling function β.

Definition 9 (Generalized weighted prefix-closure-based hybrid region objective function). *Let L be an event log over a set of activities A and let $A_s, A_d \subseteq A$ be a hybrid partition. Let $r = (m, \vec{v}, \vec{x}, \vec{y}) \in R^h_{A_s}(\overline{L})$ be a hybrid variable-based region and let β be a scaling function over $\overline{L}$, i.e. $\beta : \overline{L} \to \mathbb{R}^+$. The generalized weighted prefix-closure-based hybrid region objective function is instantiated as $c_m = \sum_{\sigma \in \overline{L}} \beta(\sigma)$, $\vec{c_v} = \sum_{\sigma \in \overline{L}} \beta(\sigma)\vec{p}_{A_s}(\sigma)$, $\vec{c_x} = \sum_{\sigma \in \overline{L}} \beta(\sigma)\vec{p}_{A_d}(\sigma)$ and $\vec{c_y} = -\vec{c_x}$, i.e.:*

$$z_\beta(r) = \sum_{\sigma \in \overline{L}} \beta(\sigma)(m + \vec{p}_{A_s}(\sigma)^\intercal \vec{v} + \vec{p}_{A_d}(\sigma)^\intercal(\vec{x} - \vec{y})) \tag{4.4}$$

Note that if we choose $\beta(\sigma) = 1$ for all $\sigma \in \overline{L}$, denoted β_1, we instantiate the generalized objective function as the objective function proposed in [3]. We denote this objective function as z_1, i.e. Equation 4.3.

To relate the behavior in a given event log to the objective function defined in Definition 9 we instantiate the scaling function β making use of the frequencies of the traces present in the event log, i.e. we let $\beta(\sigma) = \overline{L}(\sigma)$ leading to:

$$z_{\overline{L}}(r) = \sum_{\sigma \in \overline{L}} \overline{L}(\sigma)(m + \vec{p}_{A_s}(\sigma)^\intercal \vec{v} + \vec{p}_{A_d}(\sigma)^\intercal(\vec{x} - \vec{y})) \tag{4.5}$$

To assess the difference between z_1 and the newly proposed objective function $z_{\overline{L}}$, consider the Petri net depicted in Figure 2. Assume we are given some event log $L = [\langle a, b, d\rangle^5, \langle a, c, d\rangle^3]$. Let r_1 denote the hybrid variable-based region corresponding to place p_1, let r_2 correspond to p_2 and let r_3 correspond to p_3. In this case we have $z_1(r_1) = 1$, $z_1(r_2) = 1$ and $z_1(r_3) = 2$. On the other hand we have $z_{\overline{L}}(r_1) = z_{\overline{L}}(r_2) = z_{\overline{L}}(r_3) = 5 + 3 = 8$. Thus, using z_{β_L} leads to more intuitive objective values compared to using z_1 as $z_{\overline{L}}$ evaluates to the absolute number of discrete time-steps a token would remain in the corresponding place when replaying log L w.r.t. the place.

In [3] it is shown that the objective function used favors minimal regions. The proof cannot directly be adapted to hold for hybrid variable-based regions. Moreover it does not provide means to show that any arbitrary instantiation of z_β favors minimal hybrid variable-based regions. Here we show that any instantiation of the generalized weighted prefix-closure-based hybrid region objective function with some scaling function $\beta : \overline{L} \to \mathbb{R}^+$ favors minimal hybrid variable-based regions. We first show that the objective value of a non-minimal hybrid region equals the sum of the two minimal regions defining it after which we show that the given objective function maps each region to some positive real value, i.e. $rng(z_\beta) \subseteq \mathbb{R}^+$.

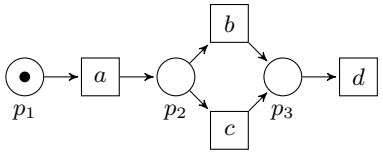

Fig. 2: A simple Petri net N with $\mathcal{L}(N) = \{\epsilon, \langle a\rangle, \langle a, b\rangle, \langle a, c\rangle\langle a, b, d\rangle, \langle a, c, d\rangle\}$.

Lemma 1 (Objective value composition of non-minimal regions). *Let L be an event log over a set of activities A and let $A_s, A_d \subseteq A$ be a hybrid partition. Let $r_1 = (m_1, \vec{v}_1, \vec{x}_1, \vec{y}_1)$, $r_2 = (m_2, \vec{v}_2, \vec{x}_2, \vec{y}_2)$ and $r_3 = (m_1+m_2, \vec{v}_1+\vec{v}_2, \vec{x}_1+\vec{x}_2, \vec{y}_1+\vec{y}_2)$ with r_1, $r_2, r_3 \in R^h_{A_s}(\overline{L})$. Let $z_\beta : R^h_{A_s}(\overline{L}) \to \mathbb{R}$ where z_β is an instantiation of the generalized weighted objective function as defined in Definition 9, then $z_\beta(r_3) = z_\beta(r_1) + z_\beta(r_2)$.*

Proof (By definition of z_β). Let us denote $z_\beta(r_3)$:

$$\sum_{\sigma \in \overline{L}} \beta(\sigma)((m_1 + m_2) + \vec{p}_{A_s}(\sigma)^\intercal(\vec{v_1} + \vec{v_2}) + \vec{p}_{A_d}(\sigma)^\intercal((\vec{x_1} + \vec{x_2}) - (\vec{y_1} + \vec{y_2})))$$

$$\sum_{\sigma \in \overline{L}} \beta(\sigma)(m_1 + \vec{p}_{A_s}(\sigma)^\intercal \vec{v_1} + \vec{p}_{A_d}(\sigma)^\intercal(\vec{x_1} - \vec{y_1}) + m_2 + \vec{p}_{A_s}(\sigma)^\intercal \vec{v_2} + \vec{p}_{A_d}(\sigma)^\intercal(\vec{x_2} - \vec{y_2}))$$

$$\sum_{\sigma \in \overline{L}} \beta(\sigma)(m_1 + \vec{p}_{A_s}(\sigma)^\intercal \vec{v_1} + \vec{p}_{A_d}(\sigma)^\intercal(\vec{x_1} - \vec{y_1})) +$$

$$\sum_{\sigma \in \overline{L}} \beta(\sigma)(m_2 + \vec{p}_{A_s}(\sigma)^\intercal \vec{v_2} + \vec{p}_{A_d}(\sigma)^\intercal(\vec{x_2} - \vec{y_2}))$$

Clearly $z_\beta(r_3) = z_\beta(r_1) + z_\beta(r_2)$. □

Lemma 1 shows that the value of z_β for a non-minimal region equals the sum of the z_β values of the two regions it is composed of. If we additionally show that z_β can only evaluate to positive values, we show that z_β favors minimal hybrid variable-based regions.

Lemma 2 (Range of z_β is strictly positive). *Let L be an event log over a set of activities A and let $A_s, A_d \subseteq A$ be a hybrid partition. Let $r = (m, \vec{v}, \vec{x}, \vec{y})$ be a non-trivial region, i.e., $r \in R^h_{A_s}(\overline{L})$. If we let z_β be any instantiation of the generalized weighted objective function as defined in Definition 9, then $z_\beta : R^h_{A_s}(\overline{L}) \to \mathbb{R}^+$.*

Proof (By showing $z_\beta(r) > 0, \forall r \in R^h_{A_s}(\overline{L})$). Let $r = (m, \vec{v}, \vec{x}, \vec{y})$ be a non-trivial hybrid variable-based region, i.e. $r \in R^h_{A_s}(\overline{L})$. Because $r \in R^h_{A_s}(\overline{L})$ we have

$$\forall_{\sigma = \sigma' \cdot \langle a \rangle \in \overline{L} \setminus \{\epsilon\}}(m + \vec{p}_{A_s}(\sigma)^\intercal \vec{v} + \vec{p}_{A_d}(\sigma')^\intercal \vec{x} - \vec{p}_{A_d}(\sigma)^\intercal \vec{y} \geq 0) \tag{4.6}$$

Note that each Parikh value of an activity $a \in A$ given a sequence σ, i.e. $\vec{p}(\sigma)(a)$, is greater or equal than the Parikh value of a, given σ's prefix, i.e.,:

$$\forall_{\sigma = \sigma' \cdot \langle a' \rangle \in \overline{L}, a \in A}(\vec{p}(\sigma)(a) \geq \vec{p}(\sigma')(a)) \tag{4.7}$$

Using Equation 4.7 we can substitute $\vec{p}_{A_d}(\sigma')^\intercal \vec{x}$ with $\vec{p}_{A_d}(\sigma)^\intercal \vec{x}$ in Equation 4.6:

$$\forall_{\sigma = \sigma' \cdot \langle a \rangle \in \overline{L} \setminus \{\epsilon\}}(m + \vec{p}_{A_s}(\sigma)^\intercal \vec{v} + \vec{p}_{A_d}(\sigma)^\intercal(\vec{x} - \vec{y}) \geq 0) \tag{4.8}$$

Combining $rng(\beta) \subseteq \mathbb{R}^+$, $\vec{p}(\epsilon) = \vec{0}$ and $m \in \mathbb{N}$ with Equation 4.8 we find $z_\beta(r) \geq 0$:

$$\sum_{\sigma \in \overline{L}} \beta(\sigma)(m + \vec{p}_{A_s}(\sigma)^\intercal \vec{v} + \vec{p}_{A_d}(\sigma)^\intercal(\vec{x} - \vec{y})) \geq 0 \tag{4.9}$$

If $m > 0$ then $\beta(\epsilon)(m+\vec{p}_{A_s}(\epsilon)^\intercal\vec{v}+\vec{p}_{A_d}(\epsilon)^\intercal(\vec{x}-\vec{y})) > 0$. Combined with Equations 4.8 and 4.9 leads to $z_\beta(r) > 0$.

Observe that if $m = 0$ then for r to be a ***non-trivial*** *hybrid variable-based region, i.e. $r \in R^h_{A_s}(\overline{L})$, either (I). $\exists a \in A_s$ s.t. $\vec{v}(a) > 0$ or (II). $\exists a \in A_d$ s.t. $\vec{x}(a) > 0$.*

(I). Let $m = 0$ and $a \in A_s$ s.t. $\vec{v}(a) > 0$. We know $\exists\sigma = \sigma' \cdot \langle a \rangle \in \overline{L}$. Because $r \in R^h_{A_s}(\overline{L})$ (using Equation 4.8) we have:

$$m + \vec{p}_{A_s}(\sigma)^\intercal\vec{v} + \vec{p}_{A_d}(\sigma)^\intercal(\vec{x} - \vec{y}) \geq 0$$

If $\sigma' = \epsilon$, we have $\vec{p}_{A_d}(\sigma) = \vec{0}$ and $\vec{p}_{A_s}(\sigma)^\intercal\vec{v} = \vec{v}(a)$ hence we deduce:

$$m + \vec{p}_{A_s}(\sigma)^\intercal\vec{v} + \vec{p}_{A_d}(\sigma)^\intercal(\vec{x} - \vec{y}) > 0$$

Combining this with Equations 4.8 and 4.9 yields $z_\beta(r) > 0$.

If $\sigma' \neq \epsilon$ we have (using Equation 4.8):

$$m + \vec{p}_{A_s}(\sigma')^\intercal\vec{v} + \vec{p}_{A_d}(\sigma')^\intercal(\vec{x} - \vec{y}) \geq 0$$

Observe that $\vec{p}_{A_s}(\sigma)^\intercal\vec{v} = \vec{p}_{A_s}(\sigma')^\intercal\vec{v} + \vec{v}(a)$, together with $\vec{p}_{A_d}(\sigma') = \vec{p}_{A_d}(\sigma)$ leads us to reformulate this to:

$$m + \vec{p}_{A_s}(\sigma)^\intercal\vec{v} - \vec{v}(a) + \vec{p}_{A_d}(\sigma)^\intercal(\vec{x} - \vec{y}) \geq 0$$

$$m + \vec{p}_{A_s}(\sigma)^\intercal\vec{v} + \vec{p}_{A_d}(\sigma)^\intercal(\vec{x} - \vec{y}) \geq \vec{v}(a)$$

Combining this with Equations 4.8 and 4.9 yields $z_\beta(r) > 0$.

(II) Let $m = 0$ and $a \in A_d$ s.t. $\vec{x}(a) > 0$. We know $\exists\sigma = \sigma' \cdot \langle a \rangle \in \overline{L}$. Because $r \in R^h_{A_s}(\overline{L})$ (using Equation 4.6) we have:

$$m + \vec{p}_{A_s}(\sigma)^\intercal\vec{v} + \vec{p}_{A_d}(\sigma')^\intercal\vec{x} - \vec{p}_{A_d}(\sigma)^\intercal\vec{y} \geq 0$$

Observe that $\vec{p}_{A_d}(\sigma)^\intercal\vec{x} = \vec{p}_{A_d}(\sigma')^\intercal\vec{x} + \vec{x}(a)$ and thus:

$$m + \vec{p}_{A_s}(\sigma)^\intercal\vec{v} + \vec{p}_{A_d}(\sigma)^\intercal(\vec{x} - \vec{y}) \geq \vec{x}(a)$$

Combining this with Equations 4.8 and 4.9 yields $z_\beta(r) > 0$. □

By combining Lemma 1 and Lemma 2 we can easily show that any instantiation of z_β favors minimal regions.

Theorem 1 (Any instantiation of z_β favors minimal regions). *Let L be an event log over a set of activities A and let $A_s, A_d \subseteq A$ be a hybrid partition. Let $r_1 = (m_1, \vec{v}_1, \vec{x}_1, \vec{y}_1)$, $r_2 = (m_2, \vec{v}_2, \vec{x}_2, \vec{y}_2)$ and $r_3 = (m_1 + m_2, \vec{v}_1 + \vec{v}_2, \vec{x}_1 + \vec{x}_2, \vec{y}_1 + \vec{y}_2)$ be three non-trivial regions, i.e., $r_1, r_2, r_3 \in R^h_{A_s}(\overline{L})$. For any $z_\beta : R^h_{A_s}(\overline{L}) \to \mathbb{R}$ being an instantiation of the generalized weighted objective function as defined in Definition 9: $z_\beta(r_3) > z_\beta(r_1)$ and $z_\beta(r_3) > z_\beta(r_2)$.*

Proof (By composition of Lemma 1 and Lemma 2). By Lemma 1 we know $z_\beta(r_3) = z_\beta(r_1) + z_\beta(r_2)$. By Lemma 2 we know that $z_\beta(r_1) > 0$, $z_\beta(r_2) > 0$ and $z_\beta(r_3) > 0$. Thus we deduce $z_\beta(r_3) > z_\beta(r_1)$ and $z_\beta(r_3) > z_\beta(r_2)$. Consequently, any instantiation of the objective function as defined in Definition 9 favors minimal regions. □

Both objective functions presented, i.e.z_1 and $z_{\overline{L}}$, are expressible in terms of the more general objective function z_β as presented in Definition 9. As we have seen the two objective functions may favors different regions. Combining an objective function with the ILP-formulation presented in Definition 8 establishes means to find Petri net places. However, solving one ILP only yields one solution and hence we need means to find a set of places, which together form a Petri net that represents the input event log L.

4.2 Finding Several Places

The most apparent technique to find multiple places is the use of causal relations within an event log. Within the context of this paper we define a causal relation as follows.

Definition 10 (Causal relation). *Let L be an event log over a set of activities A. A causal relation γ_L is a function $\gamma_L : A \times A \to \mathbb{R}$ where $\gamma_L(a, b)$ denotes the causality between activity a and b exhibited in L.*

Several approaches exist to compute causalities hence we refer to [6] for an overview of the use of causalities within different process discovery approaches. As a consequence, $\gamma_L(a, b)$ can have different meanings w.r.t. the causality between a and b. Some approaches limit $rng(\gamma_L)$ to $\{0, 1\}$ where $\gamma_L(a, b) = 1$ means that there is a causal relation between a and b whereas $\gamma_L((a, b)) = 0$ means there is no causal relation from a to b. Other approaches map $rng(\gamma_L)$ to the real-valued domain $(-1, 1)$ where a high positive $\gamma_L(a, b)$ value (close to 1) indicates a strong causal relation from a to b and a low negative value (close to -1) indicates a strong causal relation from b to a.

When adopting a causal-based ILP process discovery strategy, we try to find places which will be added in the resulting Petri net, each representing a causality found. A first step is to compute γ_L values given some causal definition. Depending on the actual meaning of the γ_L, whenever we find a causal relation from a to b (possibly because $\gamma_L(a, b) \geq \delta$, where δ is some threshold value), we enrich the constraint body for the given causal constraint as follows:

$$m = 0 \text{ and } \vec{v}(a) = 1 \text{ and } \vec{v}(b) = -1, \text{ if } a, b \in A_s$$

$$m = 0 \text{ and } \vec{x}(a) = 1 \text{ and } \vec{v}(b) = -1, \text{ if } a \in A_d, b \in A_s$$

$$m = 0 \text{ and } \vec{v}(a) = 1 \text{ and } \vec{y}(b) = 1, \text{ if } a \in A_s, b \in A_d$$

$$m = 0 \text{ and } \vec{x}(a) = 1 \text{ and } \vec{y}(b) = 1, \text{ if } a, b \in A_d$$

After the constraint body is enriched, we solve the ILP yielding a place having an optimal value for the specific objective function chosen. We repeat the procedure for every causal relation yielding a Petri net.

5 Performance

The hybrid variable-based ILP-formulation is implemented as a plug-in in the ProM-Framework (`http://www.promtools.org`) [7][2]. The plug-in allows the user to

[2]The source-code is available at `https://svn.win.tue.nl/repos/prom/Packages/HybridILPMiner/Trunk`

specify parameters of ILP-based process discovery, e.g. several different objective functions, additional constraints and causality based mining.

To test the performance of hybrid variable-based ILP process discovery we have used the basic implementation within an experimentation framework[3]. The framework allows for generating *random process models* and from these models generate *event logs*. We have generated 40 models containing a minimum of 2 activities and a maximum of 12 activities. For each model we have generated 10 event logs, each containing 5000 traces. For each log we ran three different instantiations of the process discovery formulation, one purely single variable-based, one hybrid variant and one purely dual variable-based. The distribution of event classes in A over A_s and A_d is depicted in Table 2. For the hybrid variant the size of A_s was kept constant and independent of the number of activities in A (as of $|A| \geq 3$). We ran each instantiation 10 times per log using causal relations as a process discovery strategy. For each run of an instantiation we calculated the total time spend in solving all ILPs based on the causalities present in the event log. These times have been aggregated based on the number of activities present in an event log. The results of the experiment, plotted on a logarithmic scale, are depicted in Figure 3[4].

As shown in Figure 3, the average time to solve the hybrid formulation is in-between the single and dual formulation. We additionally note the hybrid formulation to get slightly closer to the dual formulation when $|A|$ increases. This is as expected as the number of *single* variables within the hybrid formulation is constant and hence the average time of solving the ILPs within the formulation increases with respect to the single variable formulation. Figure 3 shows that reducing the number of variables used within the ILP-formulation has an impact on the average time of solving ILPs. The differences are however marginal, which is to be expected as solving ILPs is exponential by nature. Therefore the experiments show that using ILPs for the aim of process discovery in

Table 2: Different settings used in performance measurements of the hybrid variable-based ILP-formulation.

$\|A\|$	2	3	4	5	6	7	8	9	10	11	12
Purely single variable-based variant											
$\|A_s\|$	2	3	4	5	6	7	8	9	10	11	12
$\|A_d\|$	0	0	0	0	0	0	0	0	0	0	0
Hybrid variable-based variant											
$\|A_s\|$	2	3	3	3	3	3	3	3	3	3	3
$\|A_d\|$	0	0	1	2	3	4	5	6	7	8	9
Purely dual variable-based variant											
$\|A_s\|$	0	0	0	0	0	0	0	0	0	0	0
$\|A_d\|$	2	3	4	5	6	7	8	9	10	11	12

[3]All framework files and results can be found at `https://svn.win.tue.nl/repos/prom/Packages/HybridILPMiner/Tags/papers/source_files/hybrid_ilp_runtime_experiments.tar.gz`

[4]The experiments where distributed over four servers (Dell PowerEdge R520, Intel Xeon E5-2407 v2 2.40GHz, 10M Cache, 8 $\times$ 8GB RDIMM, 1600MT/s Memory), on each server a total of 10 models and corresponding logs where generated.

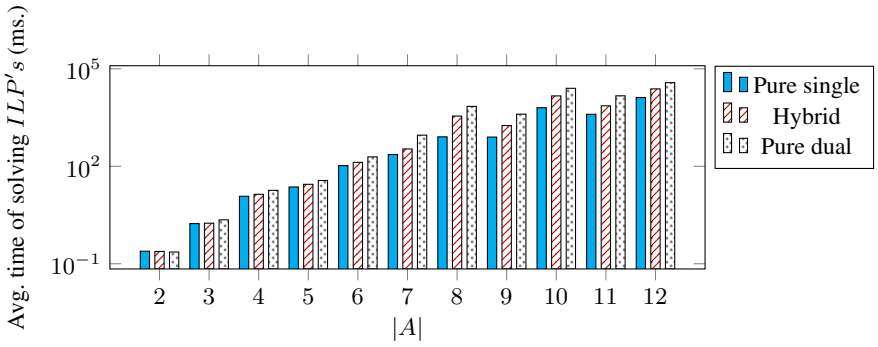

Fig. 3: Average time in milliseconds to solve the three ILP-based process discovery variants plotted on a logarithmic scale against the number of activities in the log.

a practical setting might need incorporation of more advanced techniques in order to reduce the average time of solving the ILPs significantly.

6 Related Work

The concept of hybrid variable-based regions originates from language-based region theory, which in turn originates from the area of Petri net synthesis. We identify two main branches of language-based region theory within Petri net synthesis being the *separating regions approach* [8–10] and the *minimal linear basis approach* [11, 12]. In the separating regions approach, which uses single-variable based regions, behavior not seen in a given prefix closed language is specified as being undesired. In the minimal linear basis approach, given a prefix closed language, a polyhedral cone of integer points is constructed based on dual variable-based regions. Using the cone a minimal basis of the set of regions is calculated which defines a minimal set of places to be synthesized. Both approaches try to minimize $\mathcal{L}(N) \setminus \overline{L}$, where N represents the resulting Petri net and $\overline{L}$ is a prefix-closed language. The approaches lead to Petri nets with perfect replay-fitness. Moreover precision is *maximized*. A side effect of this property is the fact that the synthesized net N scores low on both the generalization and the simplicity dimension.

In [3] a first design of an ILP-formulation was presented based on the concept of dual variable-based regions. The work presents objective function z_1 which we have further developed in this work to $z_{\overline{L}}$, and more generally, z_β. The work also focuses on formulation of several different net-types in terms of linear in-equalities which even go beyond classical Petri nets, e.g. reset- and inhibitor arcs.

An alternative approach is to use the concept of state-based regions for the purpose of process discovery [13, 14]. Within this approach an abstraction of the event log is computed in the form of a *transition system*. Regions are computed within the transition system where, like in language-based region theory, each region corresponds to a place in the resulting Petri net.

In [15] a process discovery algorithm is proposed based on the concept of *numerical abstract domains* using Parikh vectors as a basis. Based on an event log, a prefix-closed language is computed of which a convex polyhedron is approximated by means of calculating a convex hull. The convex hull is then used to compute causalities within the input log, by deducing a set of linear inequalities which represent places. The formulation used to calculate these causalities is in essence based the concept of single variable-based regions. The approach allows for finding pure Petri nets with arc weights and multiple marked places.

A multitude of other Petri net-based process discovery approaches exist. For a detailed description of these approaches we refer to [6].

7 Conclusion

We presented a new breed of language-based regions, i.e. hybrid variable-based regions, that captures the two existing region types being single and dual variable-based regions. Hybrid variable-based regions allow us to decide whether we want to use one or two variables for an activity present in the input event log. This allows us to achieve gains in terms of performance whilst maintaining the possibility to find complex (workflow) patterns. We have shown that within the hybrid variable-based ILP process discovery formulation, using only one variable per activity $a \in A$ performs optimal in terms of the average time spent in solving the ILPs constructed.

We presented a generalized weighted objective function and showed that any instantiation of the objective function leads to ILPs that favor minimal regions. As a result, practitioners may vary the scaling function within the objective function under the guarantee that the objective function favors minimal regions. We presented a log-based scaling function that exploits trace frequencies available in the input log. Using the log based scaling function within the objective function assigns an objective value to each region equal to the token throughput of the corresponding place, given the event log. Hence, as we have shown using the new objective function leads to more intuitive objective values.

As an interesting direction for future work we identify the assessment of the impact of filtering, either a-priori or within the ILP itself, on the quality dimensions of the resulting nets, i.e. are we able to leverage the perfect repaly-fitness property? Also, an assessment of the impact of decomposition techniques [16] on the performance of ILP-based approaches is an interesting direction for future work.

References

1. Aalst, W.M.P.v.d.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. 1st edn. Springer Publishing Company, Incorporated (2011)
2. Desel, J., Reisig, W.: The synthesis problem of Petri nets. Acta Inf. **33**(4) (1996) 297–315
3. Werf, J.M.E.M.v.d., Dongen, B.F.v., Hurkens, C.A.J., Serebrenik, A.: Process discovery using integer linear programming. Fundamenta Informaticae **94**(3) (2009) 387–412
4. Buijs, J.C.A.M., Dongen, B.F.v., Aalst, W.M.P.v.d.: On the role of fitness, precision, generalization and simplicity in process discovery. In Meersman, R., Panetto, H., Dillon, T.,

Rinderle-Ma, S., Dadam, P., Zhou, X., Pearson, S., Ferscha, A., Bergamaschi, S., Cruz, I.F., eds.: On the Move to Meaningful Internet Systems: OTM 2012. Volume 7565 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2012) 305–322
5. Aalst, W.M.P.v.d., Hofstede, A.H.M.t., Kiepuszewski, B., Barros, A.: Workflow patterns. Distributed and Parallel Databases **14**(1) (2003) 5–51
6. Dongen, B.F.v., Medeiros, A.K.A.d., Wen, L.: Process mining: Overview and outlook of Petri net discovery algorithms. T. Petri Nets and Other Models of Concurrency **2** (2009) 225–242
7. Dongen, B.F.v., Medeiros, A.K.A.d., Verbeek, H.M.W., Weijters, A.J.M.M., Aalst, W.M.P.v.d.: The ProM framework: A new era in process mining tool support. In Ciardo, G., Darondeau, P., eds.: Applications and Theory of Petri Nets 2005. Volume 3536 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2005) 444–454
8. Badouel, E., Bernardinello, L., Darondeau, P.: Polynomial algorithms for the synthesis of bounded nets. In Mosses, P.D., Nielsen, M., Schwartzbach, M.I., eds.: TAPSOFT '95: Theory and Practice of Software Development. Volume 915 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (1995) 364–378
9. Badouel, E., Darondeau, P.: Theory of regions. In Reisig, W., Rozenberg, G., eds.: Lectures on Petri Nets I: Basic Models. Volume 1491 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (1998) 529–586
10. Darondeau, P.: Deriving unbounded Petri nets from formal languages. In Sangiorgi, D., Simone, R.d., eds.: CONCUR'98 Concurrency Theory. Volume 1466 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (1998) 533–548
11. Bergenthum, R., Desel, J., Lorenz, R., Mauser, S.: Process mining based on regions of languages. In Alonso, G., Dadam, P., Rosemann, M., eds.: Business Process Management. Volume 4714 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2007) 375–383
12. Lorenz, R., Mauser, S., Juhas, G.: How to synthesize nets from languages - a survey. In: Simulation Conference, 2007 Winter. (Dec 2007) 637–647
13. Solé, M., Carmona, J.: Process mining from a basis of state regions. In: Applications and Theory of Petri Nets, 31st International Conference, PETRI NETS 2010, Braga, Portugal, June 21-25, 2010. Proceedings. (2010) 226–245
14. Aalst, W.M.P.v.d., Rubin, V., Verbeek, H.M.W., Dongen, B.F.v., Kindler, E., Günther, C.W.: Process mining: a two-step approach to balance between underfitting and overfitting. Software and System Modeling **9**(1) (2010) 87–111
15. Carmona, J., Cortadella, J.: Process discovery algorithms using numerical abstract domains. IEEE Trans. Knowl. Data Eng. **26**(12) (2014) 3064–3076
16. Aalst, W.M.P.v.d.: Decomposing Petri nets for process mining: A generic approach. Distributed and Parallel Databases **31**(4) (2013) 471–507