Introduction to Process Mining

R. Champagnat, M. Trabelsi, A. Hamdi et al.

2023-2024

Outline

Outline

Week 36	Introduction
Week 37	Process discovery (α-Algorithm)
Week 38	Metrics and quality of discovered models
Week 39	Raw traces/ modelled traces (case study)
Week 40	Advanced process mining algorithms
Week 41	Advanced process mining algorithms
Week 42	Conformance checking
Week 46	Decision mining in processes
Week 47	Trace clustering
Week 48	Trace profile
Week 49	Case study
Week 50	Case study defense

Evaluation

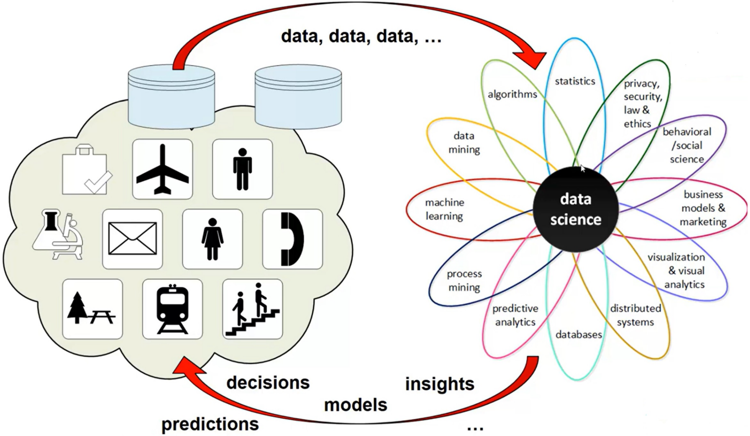
- Two Quizes (practical and theoretical)
- Case study project
  - defense: presentation (20 min) + questions (10 min)
  - report: 15 pages max
- Grades : $$\frac{(Quiz1+Quiz2+2\times defense)}{4}$$

Contributors

- Ronan Champagnat
- Marwa Trabelsi
- Noura Joudieh
- Ahmed Hamdi
- Jacques Morcos
- Mourad Rabah
- Wiem Hachicha

Outline

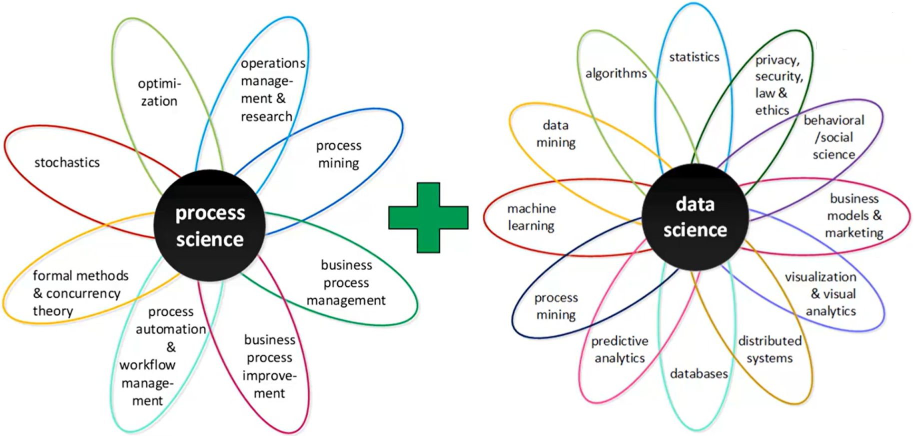
All about data



Business processes are everywhere

- **As clients, we trigger business processes**
  - Applying for a permit to build a house
  - Applying for a credit to finance property
  - Submitting an insurance claim
- **As professionals, we participate in business processes**
  - Check if the requirements for building a house are met
  - Assess the risk of granting the credit
  - Check whether a claim is covered by the insurance contract

Process science VS Data science



Process Mining: A 360 Degree Overview

Van der Aalst : Process mining in action

About Wil van der Aalst

Prof.dr.ir. Wil van der Aalst is a full professor at **RWTH Aachen University**, leading the **Process and Data Science (PADS)** group. He is also the Chief Scientist at **Celonis**, part-time affiliated with the **Fraunhofer FIT**, and a member of the Board of Governors of **Tilburg University**. He also has unpaid professorship positions at **Queensland University of Technology** (since 2003) and the **Technische Universiteit Eindhoven (TU/e)**. Currently, he is also a distinguished fellow of **Fondazione Bruno Kessler (FBK)** in Trento, deputy CEO of the **Internet of Production (IoP) Cluster of Excellence**, and co-director of the **RWTH Center for Artificial Intelligence**.

His research interests include process mining, Petri nets, business process management, workflow management, process modeling, and process analysis. Wil van der Aalst has published over **275 journal papers, 35 books (as author or editor), 630 refereed conference/workshop publications, and 85 book chapters,**.

Therory VS reality

What about information systems?

Information systems

- Set of resources and tools allowing users to search for information in a given domain.
- Business processes (i.e. a succession of activities that allow them to achieve an objective).
- Information systems are established by explicit process models that are not all clearly defined.

Unstructured processes

- Diversity of tasks, stakeholders (designers, users, managers, etc.) and other unpredictable parameters (user needs, unexpected failures or execution exceptions, etc.).
- Users can define their own processes (redundant and incomplete actions).

User model: theory vs reality

"... *if we build it, they will come* ..." (Wilson, 2003)

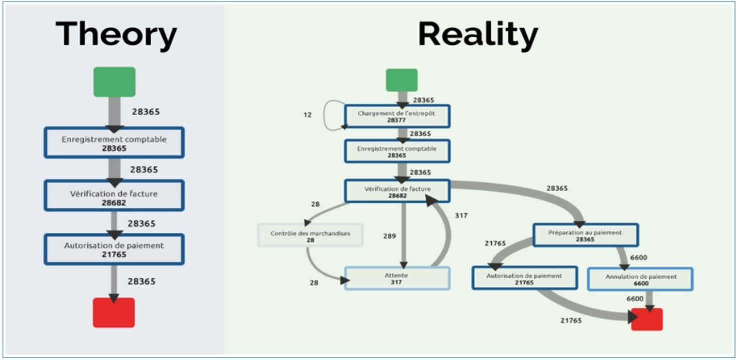
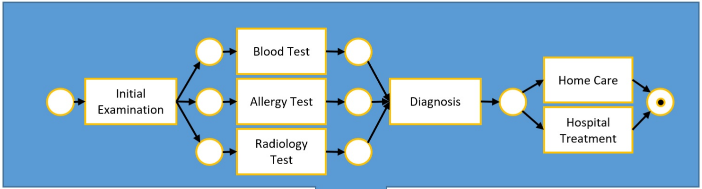


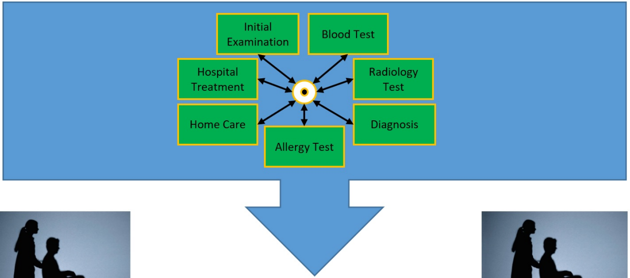
Image from `www.logpickr.com`

Hospital: theory vs reality (1)



Initial Examination
Allergy Test
Blood Test
Radiology Test
Diagnosis
Home Care

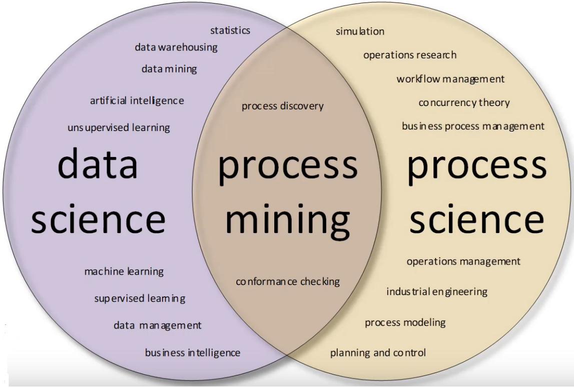
Hospital: theory vs reality (2)



Initial Examination
Allergy Test
Blood Test
Radiology Test
Diagnosis
Home Care

Initial Examination
Radiology Test
Radiology Test
Radiology Test
Radiology Test
...

Process mining as the missing link



Process Mining: A 360 Degree Overview

Outline

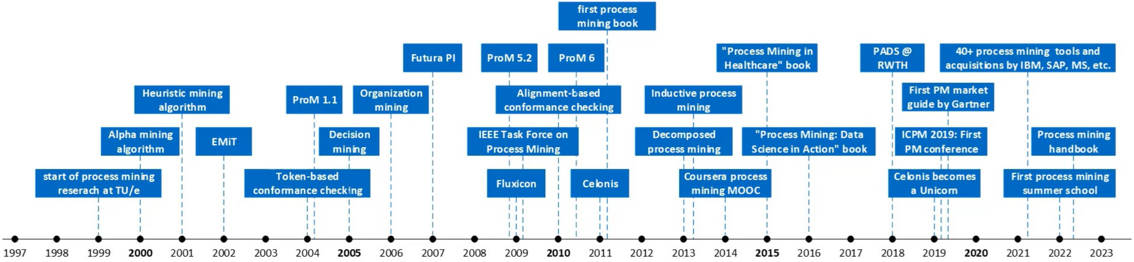
Brief history of Process Mining

- **Wil van der Aalst** pioneered the process mining field at Eindhoven University of Technology in the late 1990s.
- 2000, 1st Process Mining Algorithm (Alpha Miner)
- COOK, J. E. AND WOLF, A. L. 1995. Automating process discovery through event-data analysis. In Proceedings of the 17th International Conference on Software Engineering (Seattle, WA, April 23–30). ACM Press, New York, NY, 73–82.

Motivation

Many software process approaches and tools assume the existence of a formal process model. Unfortunately, creating a formal model for an ongoing complex process may be time-consuming, expensive, and error-prone. This is a practical impediment to the adoption of process technologies.

Timeline of Process Mining



2019, First International Conference on Process Mining

`https://icpmconference.org`

Process mining is ...

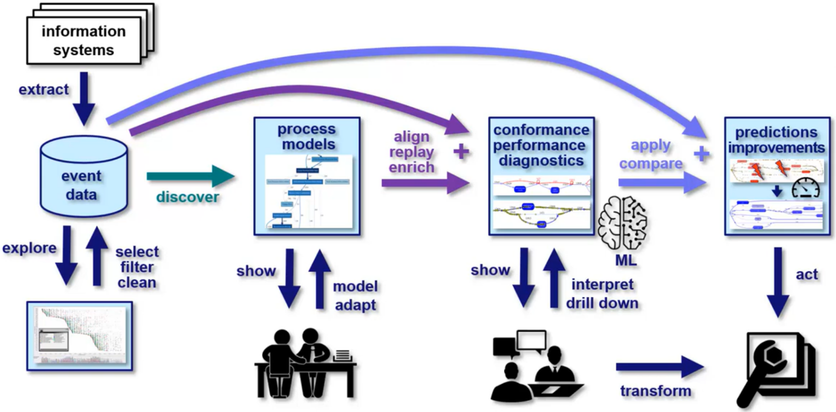
- data analysis techniques based on the process.
- event logs processing task.

Events logs are recorded data in various systems used for work (ERP, CRM, MES etc.). Analyzing event logs allows understanding

- how a certain product is manufactured?
- which itinerary, a customer goes through within a service, is identified and visualized?

.

Top down view



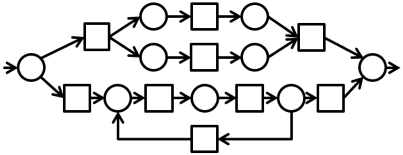
Tow main artifacts : Event logs & business processes (1)

Event logs

patient	activity	timestamp	doctor	age	cost
5781	make X-ray	23-1-2014@10.30	Dr. Jones	45	70.00
5541	blood test	23-1-2014@10.18	Dr. Scott	61	40.00
5833	blood test	23-1-2014@10.27	Dr. Scott	24	40.00
5781	blood test	23-1-2014@10.49	Dr. Scott	45	40.00
5781	CT scan	23-1-2014@11.10	Dr. Fox	45	1200.00
5833	surgery	23-1-2014@12.34	Dr. Scott	24	2300.00
5781	handle payment	23-1-2014@12.41	Carol Hope	45	0.00
5541	radiation therapy	23-1-2014@13.57	Dr. Jones	61	140.00
5541	radiation therapy	23-1-2014@13.08	Dr. Jones	61	140.00
...	...	...	...	...	...

case id | activity name | timestamp | resource | other data

Process model



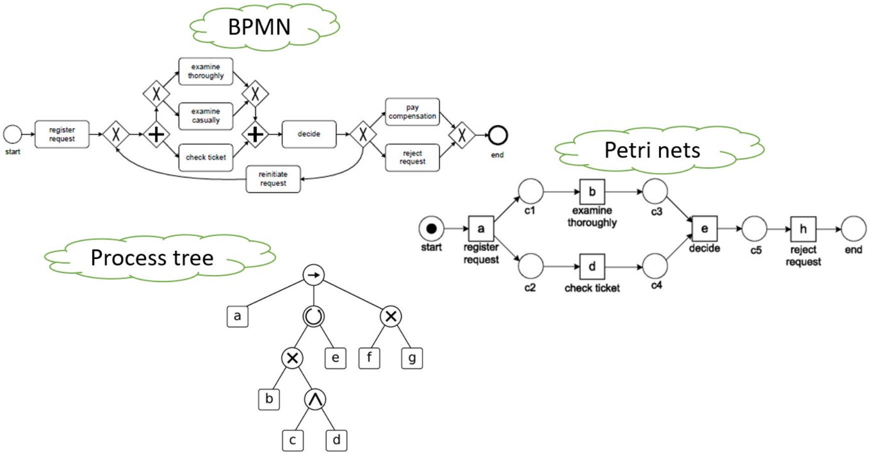
BPMN
Petri nets
DFG...

Tow main artifacts : Event logs & business processes (2)

Events are user actions in information systems that occur at a defined time. This event data is recorded in the Logs

- We all generate event data
- Phones capture data
- Internet

Tow main artifacts : Event logs & business processes (3)



Goals & Uses cases

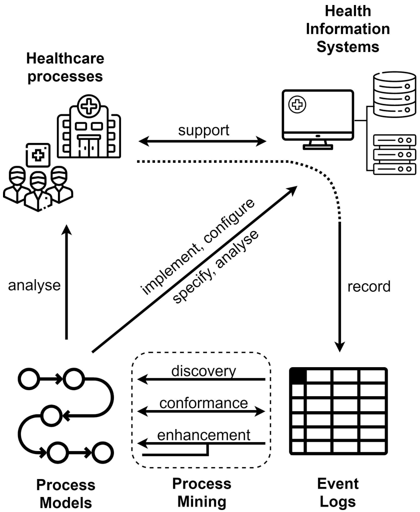
- What happened?
- Why did it happen?
- What will happen?
- What is the best that can happen?

- What is the process that people really follow?
- Where are the bottlenecks in my process?
- Where do people (or machines) deviate from the expected or idealized process?
- What about delays?

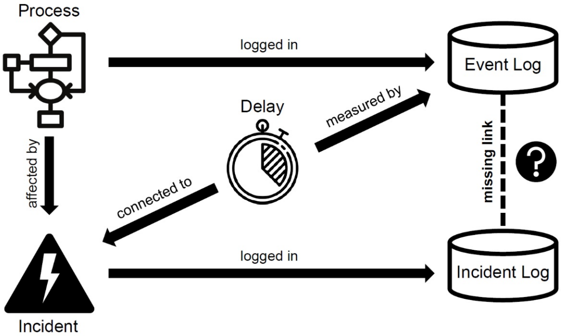
Goals & Uses cases (2)

- Company's customer service handling process
- Assessment process of Health Insurance Review & Assessment Service
- Hospital's diagnosis and treatment process
- Process of handling loan applications in banks
- Software development process
- HR management process

Goals & Uses cases (3)



Goals & Uses cases (4)



For more uses cases please take a look at this web site

https://www.tf-pm.org/resources/casestudy

Tools

- Academic
  - ProM (http://www.promtools.org/doku.php)
  - PM4Py (https://pm4py.fit.fraunhofer.de)
- Commercial (https://www.processmining-software.com)
  - Disco
  - Logpickr

Methodology

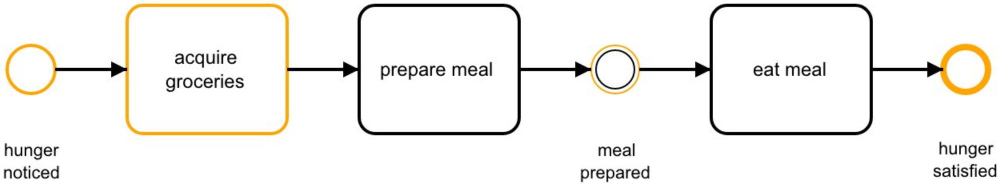
- ETL
- Data transformation
- Cleaning data
- Model discovery
- Quality measures
- Analysis
- Maintain analysis over time

Outline

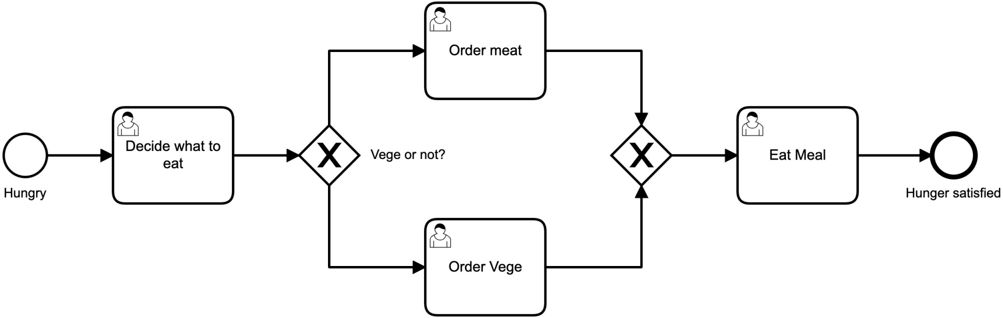
BPMN I

- BPMN (Business Process Model and Notation) is the global standard for process modelling
- BPMN is a graphical notation easily readable to represent business processes and their internal procedure
- BPMN diagram but not only (Choreography, WSBPEL, etc.)
- BPMN Elements
  - Activity
  - Event
  - Gateway
  - Flow
- References
  - https://www.bpmn.org
  - https://camunda.com/bpmn/

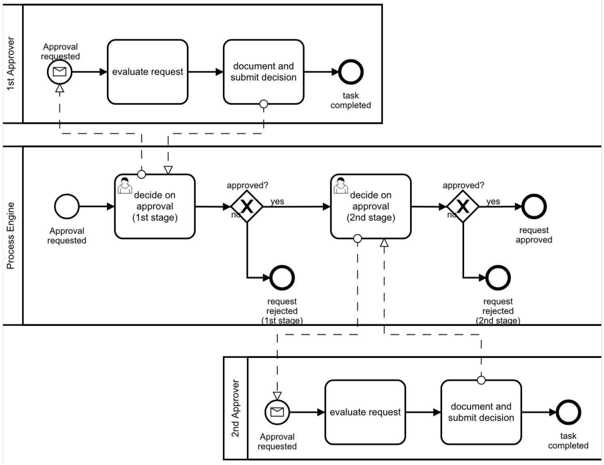
BPMN II


acquire groceries
prepare meal
eat meal
hunger noticed
meal prepared
hunger satisfied

BPMN III

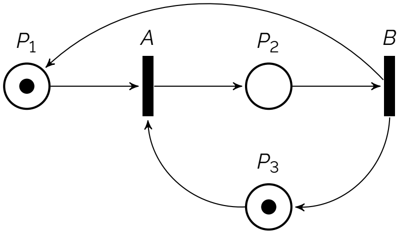

Order meat
Hungry
Decide what to eat
Vege or not?
Eat Meal
Hunger satisfied
Order Vege

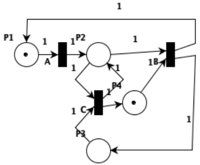
BPMN IV

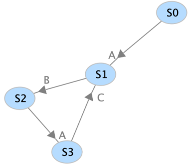

1st Approver
Approval requested
evaluate request
document and submit decision
task completed
Process Engine
Approval requested
decide on approval (1st stage)
approved?
yes
no
request rejected (1st stage)
decide on approval (2nd stage)
approved?
yes
no
request approved
request rejected (2nd stage)
2nd Approver
Approval requested
evaluate request
document and submit decision
task completed

Petri net

- Mathematical and graphical model
- Model synchronisation and resource sharing

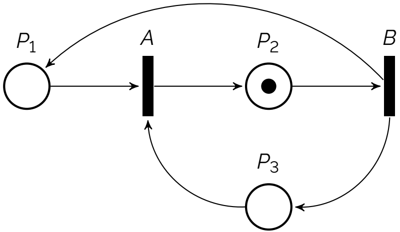


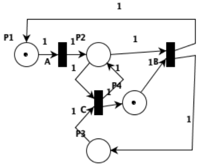


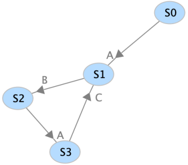


Petri net

- Mathematical and graphical model
- Model synchronisation and resource sharing

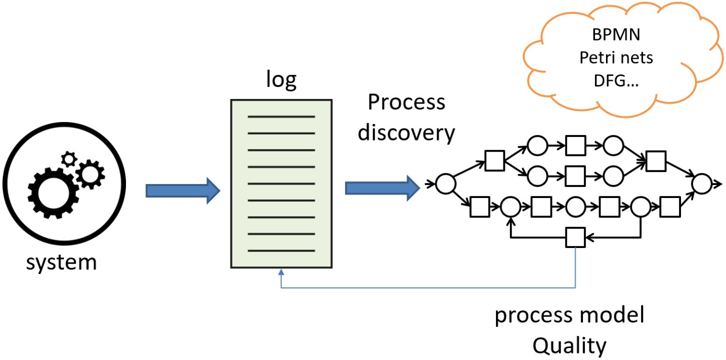






Outline

Pipeline



Why Process Mining matters ?

-> click here https://www.youtube.com/@PAFnow

What will you learn?

- Extract and analyse business processes from logs
- Transform raw data into modelled data and clean the data
- Using various process mining algorithms, extract models from logs and assess the quality of the models
- Perform conformance checking analysis
- Trace clustering

Outline

Resources

- https://www.processmining.org/home.html
- https://fluxicon.com/book/read/aboutbook/
- https://link.springer.com/chapter/10.1007/978-3-642-28108-2_19

D'ici, on voit + loin !

La Rochelle Université

univ-larochelle.fr

Process Discovery: α-Algorithm

R. Champagnat, M. Trabelsi et al.

2023-2024

Week 36	Introduction
Week 37	Process discovery (α-Algorithm)
Week 38	Metrics and quality of discovered models
Week 39	Raw traces/ modelled traces (case study)
Week 40	Advanced process mining algorithms
Week 41	Advanced process mining algorithms
Week 42	Conformance checking
Week 46	Decision mining in processes
Week 47	Trace clustering
Week 48	Trace profile
Week 49	Case study
Week 50	Case study defense

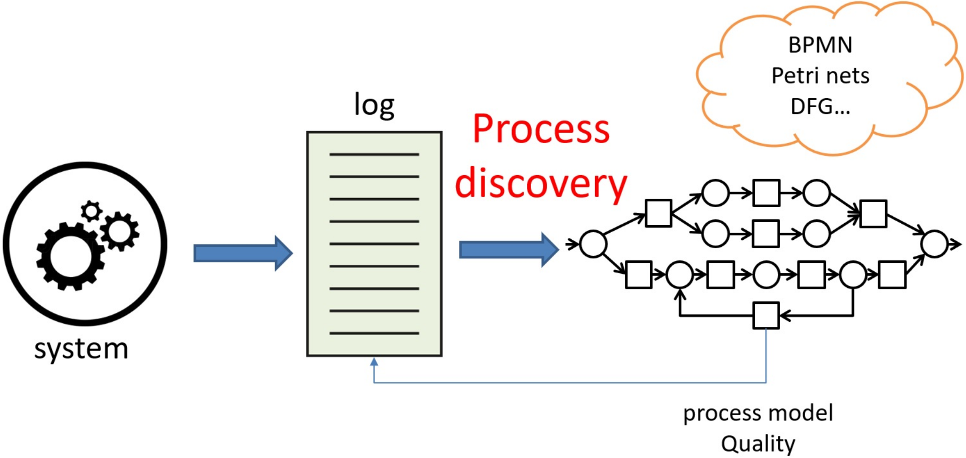
Outline

1. Preliminaries
   - Process discovery
   - Workflow nets
   - Event log
   - Process model
   - Early research
2. Process Discovery : Alpha algorithm
   - Log-based ordering relations
   - α-Algorithm
   - Limitations
3. Tools

Outline

1. Preliminaries
2. Process Discovery : Alpha algorithm
3. Tools

Preliminaries : Process discovery I

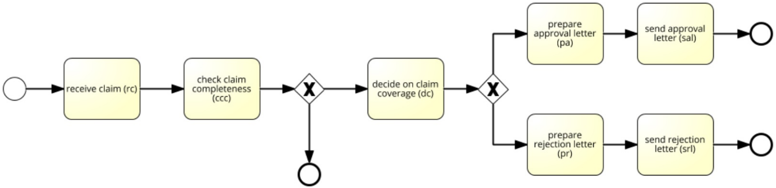


Preliminaries : Process discovery II

- Idea
  - Use traces to discover a process model
  - Hence, it models the process as it happens in reality
- Example
  - Set of trace variants
  - < rc, ccc, dc, pa, sal >, < rc, ccc, dc, pr, srl >, < rc, ccc >
- Process discovery algorithms investigate
  - Events and how events are ordered
  - Execution constraints like splits or joins
- Depending on the process discovery algorithm

Preliminaries : Process discovery III

- Process discovery by hand
  - Set of trace variants
  - < rc, ccc, dc, pa, sal >, < rc, ccc, dc, pr, srl >, < rc, ccc >
- Characterization
  - Process always starts with <rc, ccc>
  - Process can end with <pa, sal> or <pr, srl>
  - Immediately before either of these sequences, we observe <dc>
  - Process might end immediately after ccc



Preliminaries : Process discovery IV

A wide range of different process discovery algorithms have been developed

- With different assumptions and limitations
- With different notations

Algorithms

- **Alpha algorithms**
- **Heuristic Miner (HM)**
- **Inductive Miner (IM)**
- **Regions Based algorithms (SBR and ILP)**
- **Genetic Miner (GM)**
- **Fuzzy Miner (FM)**
- **etc.**

Modeling languages

- **Petri Nets**
- **Workflow Nets**
- **Process Trees**
- **Directly Follows Graphs**
- **etc.**

Preliminaries : Process discovery V

- First discovery algorithms 1995 (Cook and Wolf), 1998 (Agrawal, Gunopulos and Leymann) and 2000 (α-algorithm)
- First release of BPMN: 2006
- Base-line approach using Directly Follows Graphs (DFGs)

Workflow nets I

Process discovery algorithms use workflow nets to modelise business processes

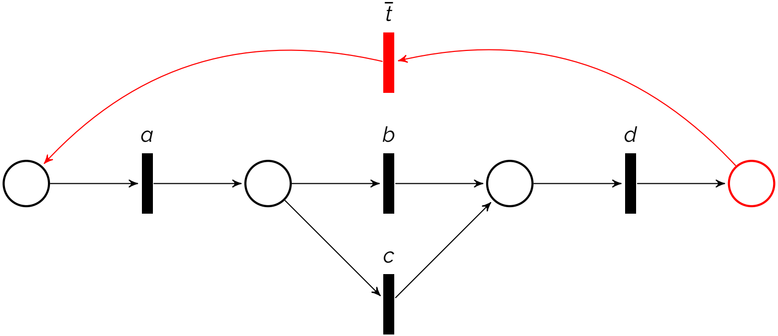
Workflow net is a restriction of Petri Net

Workflow net

Let $N = (P, T, F)$ be a Petri Net and $\bar{t}$ a fresh identifier not in $P \cup T$. N is a workflow net iff:

1. object creation : $\exists p_i \in P : \forall t \in T, \nexists Post(t, p_i)$
2. object completion : $\exists p_o \in P : \forall t \in T, \nexists Pre(p_o, t)$
3. connectedness : $\bar{N} = (P, T \cup \bar{t}, \mathcal{F} \cup \{(p_o, \bar{t}), (\bar{t}, p_i)\})$ is strongly connected.

Workflow nets II



Today we focus on α-algorithm to understand discovery issues but we will see other algorithms in the next weeks.

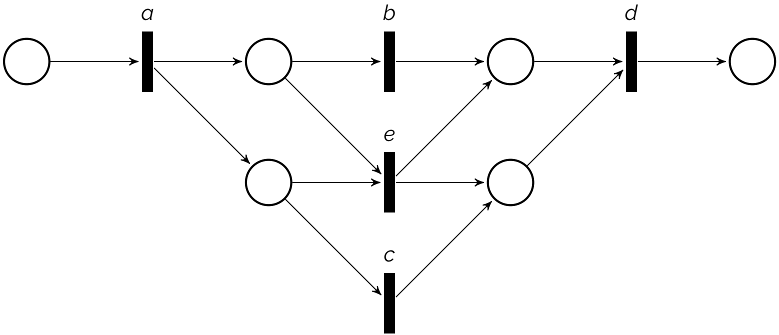
Event log

CaseId	User	Timestamp	Activity	Abbreviated
1	Roger	2016-01-12 12:34:25	Decide	a
2	Sean	2016-01-12 12:36:25	Decide	a
1	Roger	2016-01-12 12:35:26	Order Meat	b
1	Roger	2016-01-12 12:44:28	Eat Meal	d
3	Daniel	2016-01-12 12:46:26	Decide	a
3	Daniel	2016-01-12 12:50:27	Order Vege	c

- An event log is a multiset of of traces, ordered in cases, (a same case may appear multiple times). *e.g.* $L = [\langle a, b, d\rangle^2, \langle a, c, d\rangle^3]$
- A case is a sequence of activity names. *e.g.* $\langle a, b, d\rangle$

Process model I

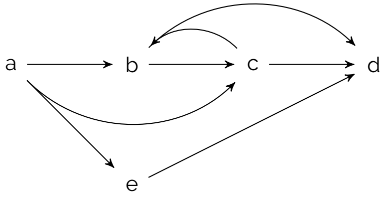
- From $L = [\langle a,b,c,d\rangle^3, \langle a,c,b,d\rangle^2, \langle a,e,d\rangle]$



- Discovered Petri Net
- Possible transition firing sequences: $\{(a,b,c,d),(a,c,b,d),(a,e,d)\}$

Process model II

- Directly-follows graph



- Possible sequences:
$\{(a, b, c, d), (a, c, b, d), (a, e, d), (a, c, d), (a, b, d), (a, b, c, b, c, d)...\}$

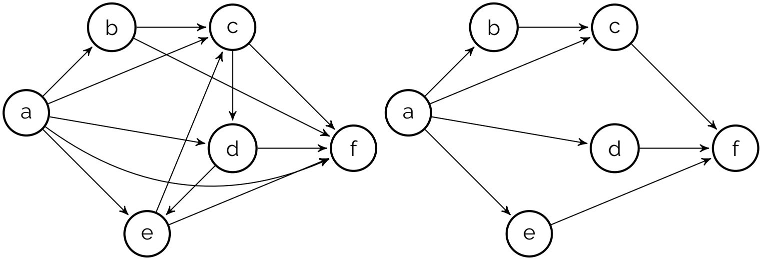
Early research I

- R. Agrawal, D. Gunopulos, and F. Leymann. Mining Process Models from Workflow Logs. In Sixth International Conference on Extending Database Technology, pages 469–483, 1998.

  1. Draw the graph of precedence constraints
  2. Remove edge that appears in both direction
  3. Remove strongly connected component
  4. Perform a graph reduction

Early research II

Let us consider $L = [\langle a, b, c, f\rangle, \langle a, c, d, f\rangle, \langle a, d, e, f\rangle, \langle a, e, c, f\rangle]$



Early research III

- J.E. Cook and A.L. Wolf. Discovering Models of Software Processes from Event-Based Data. ACM Trans- actions on Software Engineering and Methodology, 7(3):215–249, 1998. They describe three methods for process discovery:
  - using neural networks
  - purely algorithmic approach
  - Markovian approach

They propose specific metrics (entropy, event type counts, periodicity, and causality) and use these metrics to discover models out of event streams.

Early research IV

- W. van der Aalst, T. Weijters and L. Maruster, "Workflow mining: discovering process models from event logs," in IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 9, pp. 1128-1142, Sept. 2004, doi: 10.1109/TKDE.2004.47.

Outline

Log-based ordering relations

- Analyze causal dependencies of activities in the log (*e.g.* if an activity is always followed by another activity it is likely that there is a causal relation between both activities)
- We will consider the forth following relations between any activities a_1 and a_2:
  1. *Direct succession* : $a_1 > a_2$ if there is a trace such that a_1 is immediately followed by a_2 in a log;
  2. *Causality* : $a_1 \rightarrow a_2$, if $a_1 > a_2$ and $a_2 \not> a_1$;
  3. *Parallel* : $a_1 \| a_2$, if $a_1 > a_2$ and $a_2 > a_1$;
  4. *Choice* : $a_1 \# a_2$, if $a_1 \not> a_2$ and $a_2 \not> a_1$.

Ordering relations example

From the following logs $L = [\langle a, b, c, d\rangle^3, \langle a, c, b, d\rangle^2, \langle a, e, d\rangle]$, we can extract the following relations

- *Direct succession relations* ($>$):
  - $a > b$, $a > c$, $a > e$, $b > c$, $b > d$, $c > b$, $c > d$, $e > d$;
- *Causality* ($\rightarrow$):
  - $a \rightarrow b$, $a \rightarrow c$, $a \rightarrow e$, $b \rightarrow d$, $c \rightarrow d$, $e \rightarrow d$;
- *Parallel* ($\|$):
  - $b \| c$, $c \| b$;
- *Choice* ($\#$) :
  - $b \# e$, $e \# b$, $c \# e$, $e \# c$, $a \# d$, $d \# a$.

α-algorithm I

Formallly:

1. $T_L = \{t \in T | \exists_{\sigma \in L} t \in \sigma\}$
2. $T_I = \{t \in T | \forall_{\sigma \in L} t = first(\sigma)\}$
3. $T_O = \{t \in T | \forall_{\sigma \in L} t = last(\sigma)\}$
4. $X_L = \{(A, B) | A \subseteq T_L \wedge A \neq \varnothing \wedge B \subseteq T_L \wedge B \neq \varnothing \wedge \forall_{a \in A} \forall_{b \in B} a \rightarrow_L b \wedge \forall_{a_1, a_2 \in A} a_1 \#_L a_2 \wedge \forall_{b_1, b_2 \in B} b_1 \#_L b_2\}$
5. $Y_L = \{(A, B) \in X_L | \forall_{A', B' \in X_L} A \subseteq A' \wedge B \subseteq B' \Rightarrow (A, B) = (A', B')\}$
6. $P_L = \{p_{(A,B)} | (A, B) \in Y_L\} \cup \{i_L, o_L\}$

α-algorithm II

7. $F_L = \{(a, p_{(A,B)}) | (A,B) \in Y_L \wedge a \in A\} \cup \{(p_{(A,B)}, b) | (A,B) \in Y_L \wedge b \in B\} \cup \{(i_L, t) | t \in T_I\} \cup \{(t, o_L) | t \in T_o\}$

8. $\alpha(L) = (P_L, T_L, F_L)$

α-algorithm III

In detail:

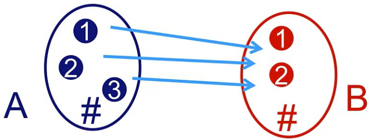
1. $T_L = \{t \in T | \exists_{\sigma \in L} t \in \sigma\}$. Extract transitions names (an activity is a transition)
2. $T_I = \{t \in T | \forall_{\sigma \in L} t = \mathit{first}(\sigma)\}$. Fix the set of start activity
3. $T_O = \{t \in T | \forall_{\sigma \in L} t = \mathit{last}(\sigma)\}$. Fix the set of end activity

α-algorithm IV

4. $X_L = \{(A,B) | A \subseteq T_L \wedge A \neq \varnothing \wedge B \subseteq T_L \wedge B \neq \varnothing \wedge \forall_{a \in A} \forall_{b \in B} a \rightarrow_L b \wedge \forall_{a_1,a_2 \in A} a_1 \#_L a_2 \wedge \forall_{b_1,b_2 \in B} b_1 \#_L b_2\}$.

Find pairs (A, B) of sets of activities such that:

- Every element $a \in A$ and every element $b \in B$ are causally related (*i.e.* $a \rightarrow b$)
- All elements in A are independent ($a_1 \# a_2$), and all elements in B are independent ($b_1 \# b_2$).



α-algorithm V

5. $Y_L = \{(A,B) \in X_L | \forall_{A',B' \in X_L} A \subseteq A' \wedge B \subseteq B' \Rightarrow (A,B) = (A',B')\}$. Delete non-maximal pairs (A,B) from X_L, For instance: let us take $a, b, c \in T$ with $a \rightarrow b, a \rightarrow c$ and $b \# c$ then $(\{a\},\{b\})$ in X and $(\{a\},\{b,c\})$ also. The goal is to reduce the number of places to keep the ones that connect the maximum of transitions (here $(\{a\},\{b,c\})$).

6. $P_L = \{p_{(A,B)} | (A,B) \in Y_L\} \cup \{i_L, o_L\}$. Determine the place set: each element (A,B) of Y_L is a place. And add source and target places.

7. $F_L = \{(a, p_{(A,B)}) | (A,B) \in Y_L \wedge a \in A\} \cup \{(p_{(A,B)}, b) | (A,B) \in Y_L \wedge b \in B\} \cup \{(i_L, t) | t \in T_I\} \cup \{(t, o_L) | t \in T_o\}$. Determine the flow relation by connecting each place $p_{(A,B)}$ with each element a of its set A of source transitions and with each element b of its set B of target transitions. In addition, draw an arc from the source place i_L to each start transition $t \in T_I$ and an ark from each end transition $t \in T_o$ to the sink place o_L

α-algorithm VI

8. $\alpha(L) = (P_L, T_L, F_L)$. The discovered Petri Net.

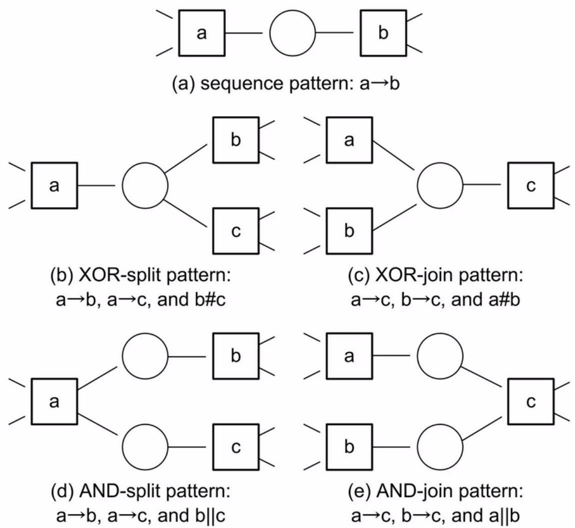
The whole concept

- Find pairs that are maximal (step 5).
- If two activities follow there is a place in between.
- A place defines a local constraint

A place is a constraint

- If we have a sequential pattern $a \rightarrow b$, the place between the transition a and b specifies that a and b should happen the same number of times and b should be executed after a.

α-algorithm VII



Example I

$$L = [\langle a,b,c,d\rangle^3, \langle a,c,b,d\rangle^2, \langle a,e,d\rangle]$$

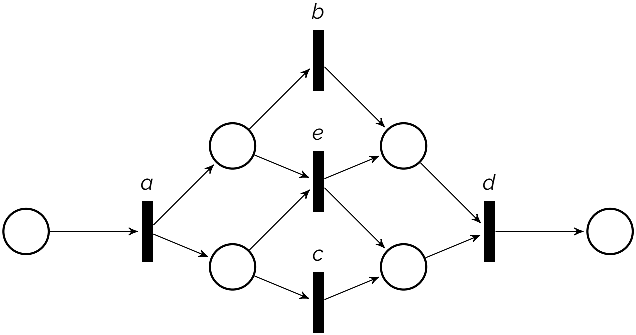
1. a, b, c, d, e
2. a
3. d
4.

	a	b	c	d	e
a	$\#$	$\rightarrow$	$\rightarrow$	$\#$	$\rightarrow$
b	$\leftarrow$	$\#$	$\parallel$	$\rightarrow$	$\#$
c	$\leftarrow$	$\parallel$	$\#$	$\rightarrow$	$\#$
d	$\#$	$\leftarrow$	$\leftarrow$	$\#$	$\leftarrow$
e	$\leftarrow$	$\#$	$\#$	$\rightarrow$	$\#$

$$X_L = \{(\{a\},\{b\}),(\{a\},\{c\}),(\{a\},\{e\}),(\{a\},\{b,e\}),(\{a\},\{c,e\}),(\{b\},\{d\}), (\{c\},\{d\}),(\{e\},\{d\}),(\{b,e\},\{d\}),(\{c,e\},\{d\})\}$$

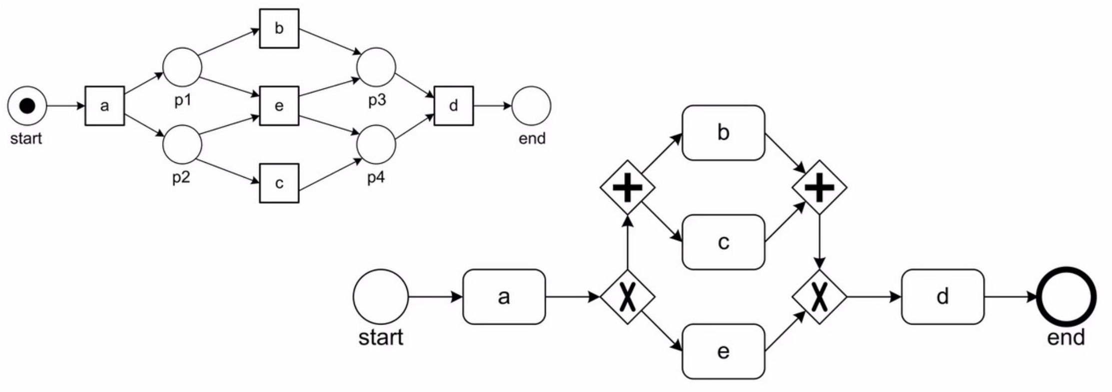
Example II

5. $Y_L = \{(\{a\},\{b,e\}),(\{a\},\{c,e\}),(\{b,e\},\{d\}),(\{c,e\},\{d\})\}$



6.

Example III



Limitations

- The discovered model is not optimal (implicit places)
- Cannot discover loops (length 1 and more)
- Non-local dependencies

Outline

1. Preliminaries
2. Process Discovery : Alpha algorithm
3. Tools

Tools

- ProM (`http://promtools.org/`)
- PM4PY (`https://pm4py.fit.fraunhofer.de`)


D'ici, on voit + loin !
La Rochelle Université
univ-larochelle.fr

Quality Criteria

R. Champagnat, M. Trabelsi, A. Hamdi et al.

2023-2024

Week 36 Introduction

Week 37 Process discovery (α-Algorithm)

Week 38 Metrics and quality of discovered models

Week 39 Raw traces/ modelled traces (case study)

Week 40 Advanced process mining algorithms

Week 41 Advanced process mining algorithms

Week 42 Conformance checking

Week 46 Decision mining in processes

Week 47 Trace clustering

Week 48 Trace profile

Week 49 Case study

Week 50 Case study defense

Outline

Outline

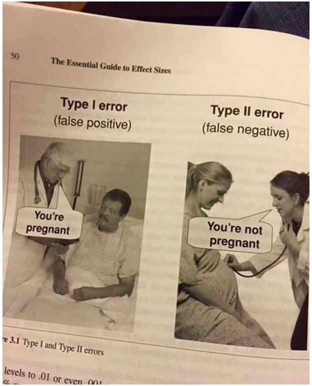
Introduction I

- To evaluate is to measure the reliability of a tool
- Evaluation depends on the phenomena to be assessed
- The approach requires a ground truth
- Assessment of results depends on the intended application
- Evaluation must be reproducible

Introduction II

- In a classification system (binary)
  - True positive
  - True negative
  - False positive
  - False negative
- while 0 is the negative class and 1 is the positive class

Real	Predicted	
0	0	**TN**
0	1	**FP**
1	0	**FN**
1	1	**TP**



Introduction III

Let $p_1, p_2, p_3, p_4, p_5, p_6, p_7$ be a set of data representing pebbles and A and B two classes with :

- A (pebbles with gold nuggets)
- B (pebbles of no interest)

Ground truth

p_1	A
p_2	A
p_3	A
p_4	A
p_5	B
p_6	B
p_7	B

Predictions

p_1	A
p_2	B
p_3	A
p_4	B
p_5	B
p_6	A
p_7	B

Confusion matrix

GT \ pred	A	B
A	p_1, p_3 (**TP**)	p_2, p_4 (**FN**)
B	p_6 (**FP**)	p_5, p_7 (**TN**)

Introduction IV

- Precision : rate of correct answers
  TRUE POSITIVE VS FALSE POSITIVE
  "Among the positive predictions, how many are really positive?"
- Recall : rate of answers found
  TRUE POSITIVE VS FALSE NEGATIVE
  "Among the real positives, how many are predicted positive?"

- Noise : rate of incorrect answers
- Silence : rate of forgotten answers

- Noise = $1 - \text{precision} \rightarrow$ errors of type I
- Silence = $1 - \text{recall} \rightarrow$ errors of type II

La Rochelle Université

Introduction V

- Precision = $\frac{\text{number of relevant items found}}{\text{number of items found}}$
- $\rightarrow P = \frac{\text{TP}}{\text{TP + FP}}$
- Recall = $\frac{\text{number of relevant items found}}{\text{number of relevant items}}$
- $\rightarrow R = \frac{\text{TP}}{\text{TP + FN}}$

F-measure :

$$F_\beta = (1+\beta^2)\frac{PR}{(\beta^2 P) + R}$$

- $\beta = 1 \rightarrow$ balance between P and R
- $\beta < 1 \rightarrow$ P is favored
- $\beta > 1 \rightarrow$ R is favored

$\rightarrow$ What about process mining?

Introduction VI

- Venn Diagram[1]

Software Process Validation • 151

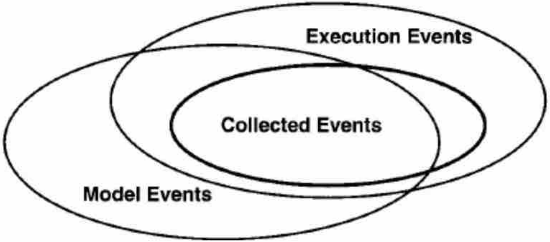


Fig. 1. Venn diagram of event types.

- Is the discovered model a correct reflection of the real process?
- what is the quality of the discovered model?

Introduction VII

Classification approaches (using confusion matrix) define:

- TP: traces possible in model and also possible in real process.
- TN: traces not possible in model and also not possible in real process.
- FP: traces possible in model but not possible in real process.
- FN: traces not possible in model but possible in real process.

Cannot be used since the identified model generates infinite sequences and log only contains a subset of all potential traces.
$\Rightarrow$ Need for defining specific measures

[1] J.E. Cook and A.L. Wolf. Software Process Validation: Quantitatively Measuring the Correspondence of a Process to a Model. ACM Transactions on Software Engineering and Methodology (TOSEM), 8:147–176, April 1999.

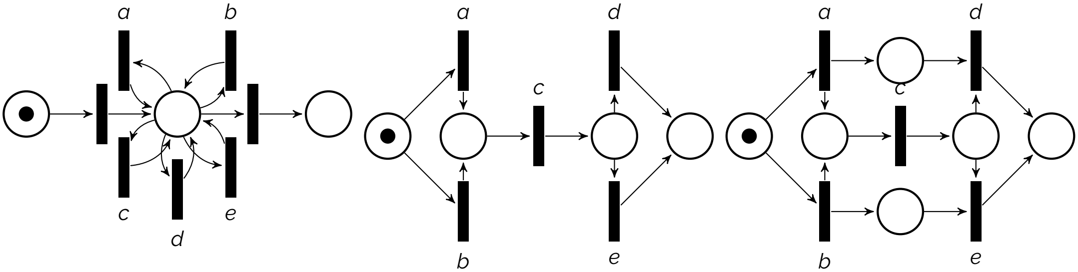
Outline

What is the best model? I

- How good is my model?
- There are many different process discovery algorithms available
- Many discovery algorithms build on parameters and, therefore, can produce different models
- How can we assess whether a resulting model is "good"?
- We can build on the notion of underfitting and overfitting from machine learning

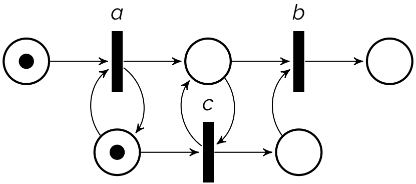
What is the best model? II

Let us consider the following log $L = [\langle a, c, d \rangle^{99}, \langle b, c, e \rangle^{85}]$ we can deduce the candidate models:

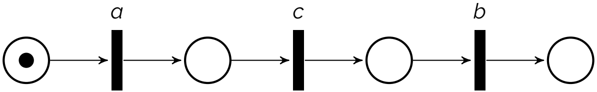


And with logs $L = [\langle a, c, d \rangle^{99}, \langle a, c, e \rangle^{50}, \langle b, c, e \rangle^{85}, \langle b, c, d \rangle^{48}]$ or $L = [\langle a, c, d \rangle^{99}, \langle a, c, e \rangle^{1}, \langle b, c, e \rangle^{85}, \langle b, c, d \rangle^{2}]$?

What is the best model? III

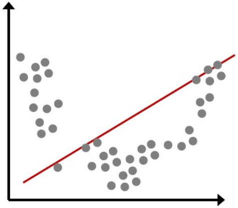


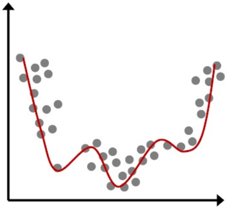
Or



What is underfitting and overfitting?

In machine learning, we often fit a model to training data for the purpose of prediction





- **Underfitting**
- Large distance from line to most data points
- The shape of the model and the data are very different

- **Overfitting**
- Low distance from line to most data points
- The shape of the model and the data are very similar

Outline

1 Introduction

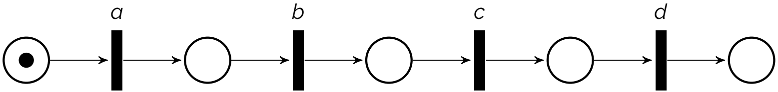
2 Why using Quality Criteria?

3 Quality Criteria for Process Dicovery

4 Initial Measures

Notion of Overfitting

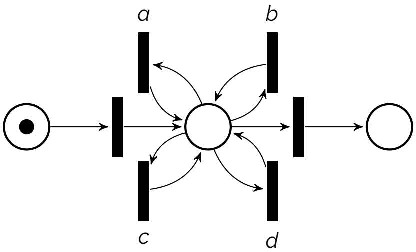
Allows only the discovered behaviour (the next trace will not fit)



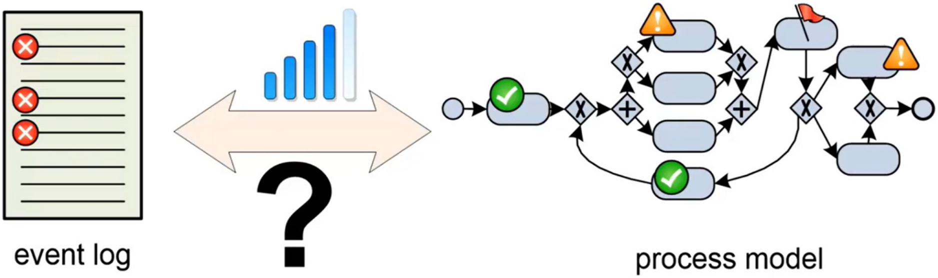
Notion of Underfitting

Underfitting

Allows too much behaviour



Quality Criteria in Process Mining

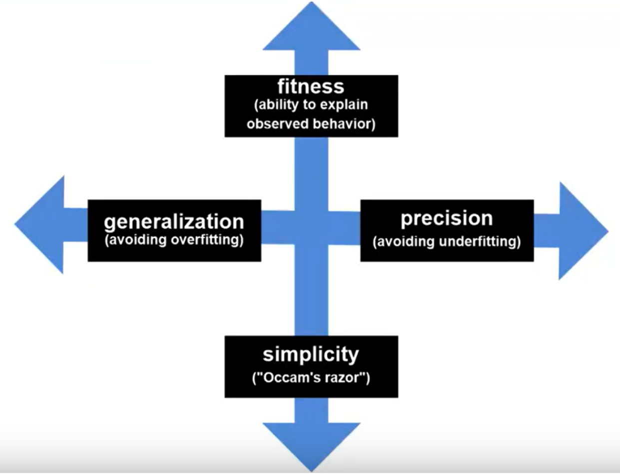


Four Quality Criteria for Process Mining I

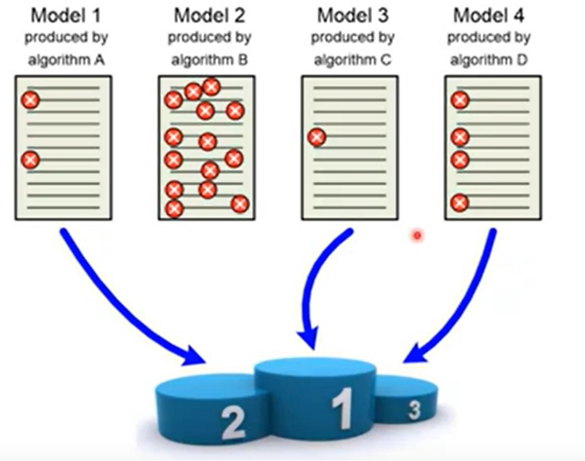
Buijs Joos et al. (2012). **On the Role of Fitness, Precision, Generalization and Simplicity in Process Discovery**.

- **Fitness**: ability to explain observed behaviour
- **Precision**: (avoid underfitting): the discovered model should not allow for behavior completely unrelated to what was seen in the event log.
- **Generalisation**: (avoid overfitting): the discovered model should generalize the example behavior seen in the event log.
- **Simplicity**: complexity and specificity of the model

Four Quality Criteria for Process Mining II



Best process discovery algorithm



Fitness I

Indicates how much the observed behaviour in the log is captured by the process model

- In general -> $f = \frac{number\_of\_traces\_captured\_by\_the\_model}{number\_of\_traces\_in\_the\_log}$
- **Comparing footprints**
- **Token-Based Replay**
- **Alignment**

La Rochelle Université

Fitness : Comparing footprints

Compare the footprints of log (L) and possible traces of the model (M).

L	a	b	c	d	e
a	$\#$	$\rightarrow$	$\rightarrow$	$\#$	$\rightarrow$
b	$\leftarrow$	$\#$	$\parallel$	$\rightarrow$	$\#$
c	$\leftarrow$	$\parallel$	$\#$	$\rightarrow$	$\#$
d	$\#$	$\leftarrow$	$\leftarrow$	$\#$	$\leftarrow$
e	$\leftarrow$	$\#$	$\#$	$\rightarrow$	$\#$

M	a	b	c	d	e
a	$\#$	$\rightarrow$	$\#$	$\#$	$\rightarrow$
b	$\leftarrow$	$\#$	$\parallel$	$\rightarrow$	$\#$
c	$\leftarrow$	$\parallel$	$\#$	$\rightarrow$	$\#$
d	$\#$	$\leftarrow$	$\leftarrow$	$\#$	$\parallel$
e	$\leftarrow$	$\#$	$\#$	$\rightarrow$	$\#$

$$f = 1 - \frac{number\_of\_different\_elements}{number\_of\_elements}$$

Fitness : Token-Based Replay I

- Given an **event log** and a **Petri net**, token based-replay takes each trace in the log in isolation and fire transitions sequentially according to the ordering of events in the trace.
- If a transition should be fired according to an event in a trace but it is not enabled, **missing tokens** are added to enable the transition.
- All added tokens are recorded.
- Together with the number of **remaining tokens** left after all traces are replayed, the amount of added tokens is used to measure conformance between the log and the net.

Fitness : Token-Based Replay II

While replay progresses, we count the number of tokens that had to be created artificially (i.e., the transition belonging to the logged event was not enabled and therefore could not be successfully executed) and the number of tokens that were left in the model,which indicate that the process was not properly completed[2]

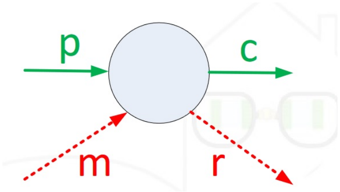
$$f = \frac{1}{2}(1 - \frac{\sum_{i=1}^{k} n_i m_i}{\sum_{i=1}^{k} n_i ci}) + \frac{1}{2}(1 - \frac{\sum_{i=1}^{k} n_i r_i}{\sum_{i=1}^{k} n_i pi}) \quad (1)$$

Where:

- i is the log trace index,
- n_i is the number of process instances combined into the current trace,
- c_i is the number of consumed tokens,

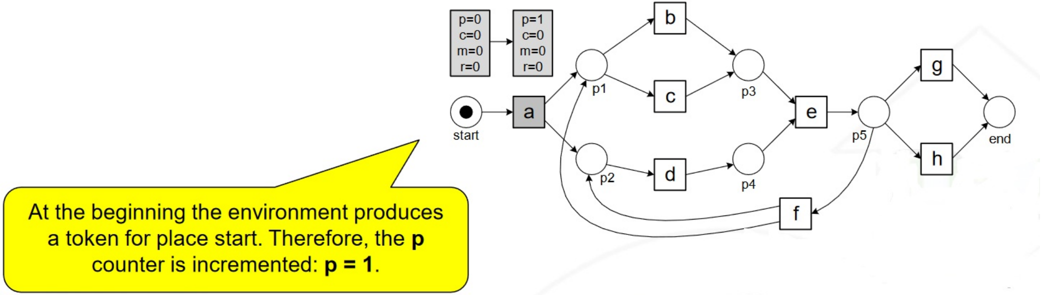
Fitness : Token-Based Replay III

- p_i is the number of produced tokens during log replay of the current trace,
- m_i is the number of missing tokens,
- r_i is the number of remaining tokens.

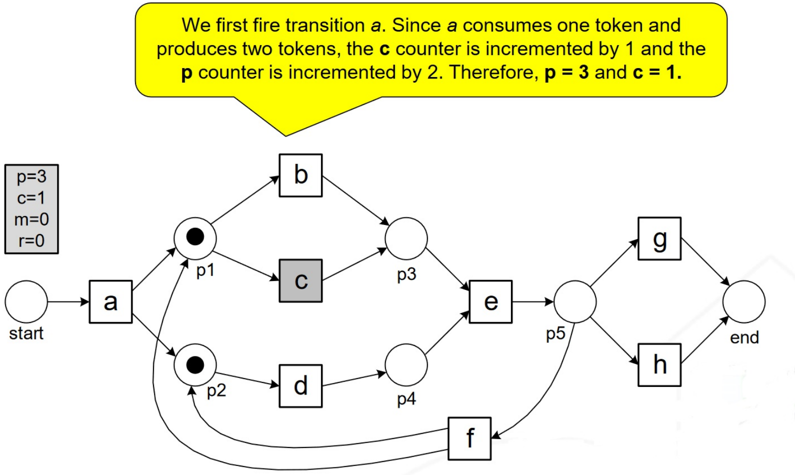


Fitness : Token-Based Replay IV

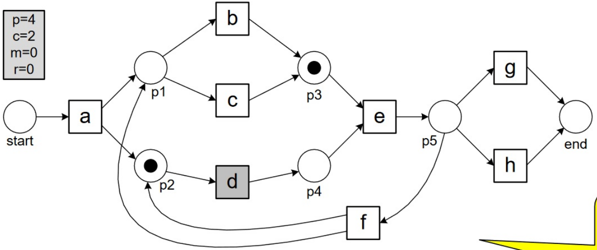
Let's replay the trace "*acdeh*" on this discovered model. Initially, $p = c = 0$ and all places are empty.



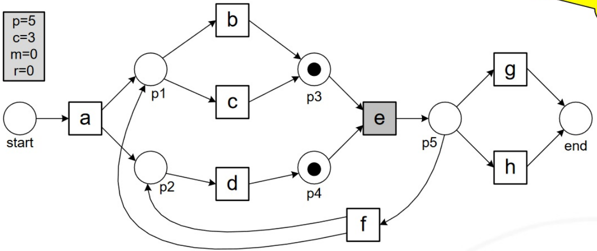
Fitness : Token-Based Replay V



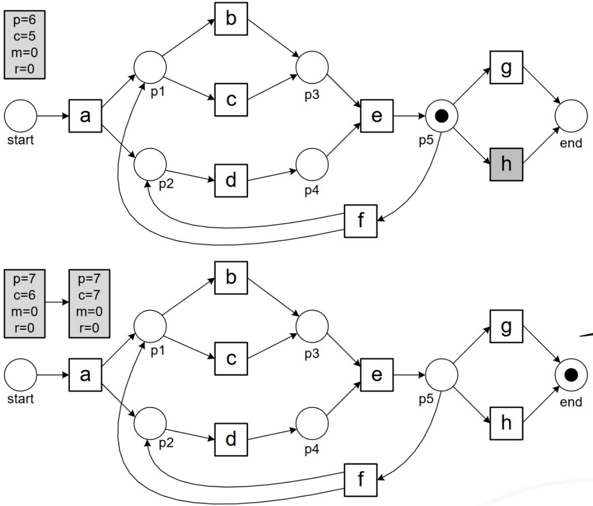
Fitness : Token-Based Replay VI



Then we replay the second event *c* and the third event *d*.
p = 5 and **c = 3**



Fitness : Token-Based Replay VII



At the end, the environment consumes one token from place end. Hence, **p = c = 7**

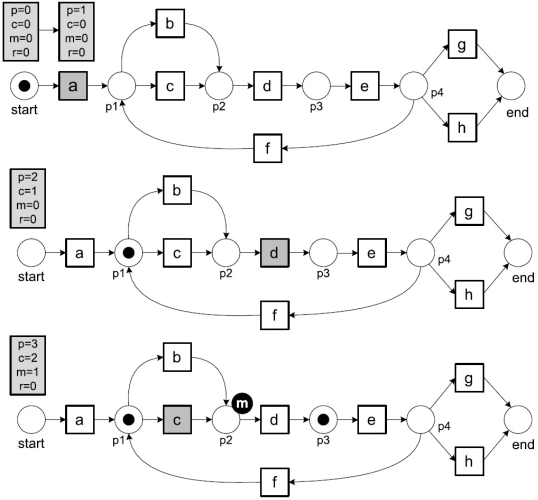
Fitness : Token-Based Replay VIII

So for the first example the fitness value will be :

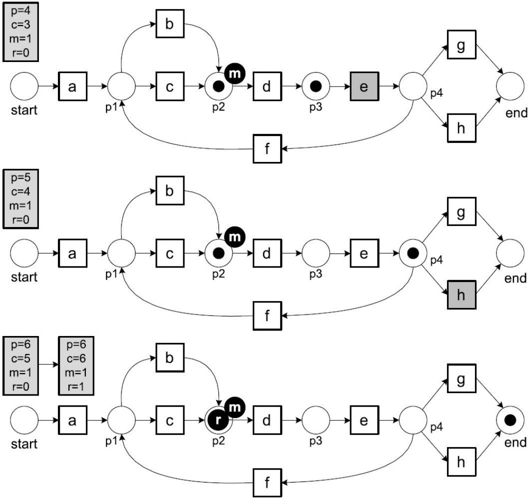
$$f = \frac{1}{2}(1 - \frac{0}{7}) + \frac{1}{2}(1 - \frac{0}{7}) = 1 \tag{2}$$

Let's replay another trace "*adceh*" on this discovered model. Initially, $p = c = 0$ and all places are empty.

Fitness : Token-Based Replay IX

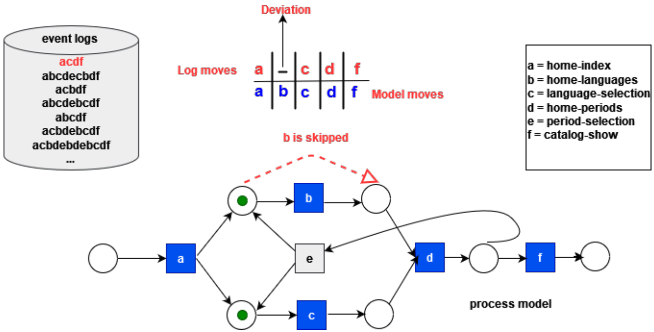


Fitness : Token-Based Replay X



[2] Rozinat et al. (2008). Conformance checking of processes based on monitoring real behavior. Information Systems

Fitness : Computation based on alignment-based algorithms



Precision I

Precision

- Does the model allow for traces that are not in the event log?
- Determining how many traces from the model are not part of the event log.
- This is not always straightforward since models often allow for an infinite number of traces

In general, Precision quantifies the fraction of the behavior allowed by the model which is not seen in the event log [3]

$$Precision(L, M) = \frac{1}{|E|} \sum_{e \in E} \frac{|en_L(e)|}{|en_M(e)|} \tag{3}$$

where:

- $|E|$ is the number of events in the log L (the number of lines of the log)

Precision II

- e is an event in the log (one line)
- $en_L(e)$ is the set of activities (event types) enabled in the event logs
- $en_M(e)$ be the set of activities enabled in the model
- $en_L(e) \subseteq \mathrm{en}_M(e)$ because the event log is perfectly fitting. Therefore, $0 < precision(L, M) \leq 1$.
- Precision is 1 if all the possible behaviors allowed in the model are observed in the log.
- If the model allows for much more behavior than observed, then $precision(L, M) \leq 1$.

Precision III

Munoz-Gama et al. (2011). **Enhancing precision in Process Conformance: Stability, confidence and severity.**

- The precision metric avoids enumerating all the possible states.
- Requires to calculate the prefix automaton based on the log

Precision IV

# Instances	Log Traces
1435	A B D E A
946	A C D G H F A
764	A C G D H F A
54	A C G H D F A
1	A C D G G H F A

(a) Log L_1

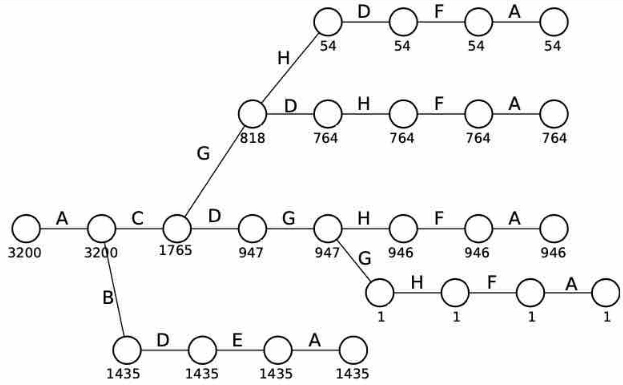
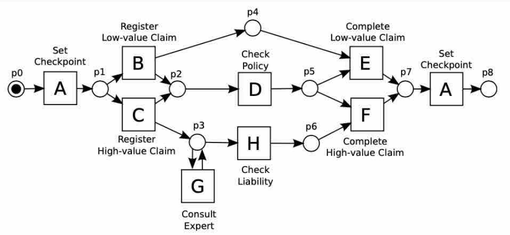


Fig. 2. Prefix Automaton

Precision V

- For each state identify the escape states (enabled transition in the model and note in the log when replaying the log)



(b) Model M_1

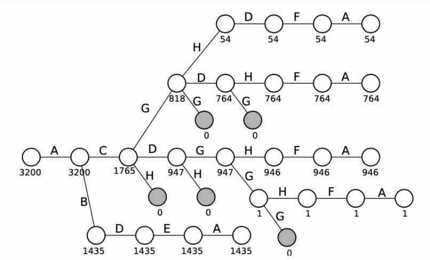


Fig. 3. Extended Prefix Automaton

Precision VI

$$etc_p = 1 - \frac{\sum_{for\_each\_state}(number\_of\_escaping\_states \times occurence\_of\_the\_prefix)}{\sum_{for\_each\_state}(number\_of\_available\_states \times occurence\_of\_the\_prefix)} \quad (4)$$

- For more details please see this Helpers' presentation Precision helpers
- Both metrics require the log to fit the model.

[3] Aalst 2016, Process mining: data science in action

Generalization and Simplicity

Generalization

Assesses the extent to which the resulting model will be able to reproduce future behavior of the process.

Simplicity

Quantify the complexity of a process model

Based on complexity measures of a process model

Combination of measures

F-measure

$$F - measure = \frac{2 * F * P}{F + P}$$

- A proper process model must find a balance between quality criteria.
- It has been shown that Fitness and Precision are linked. A small amount of behaviors (event logs) leads to a decrease in Fitness and an increase in Precision

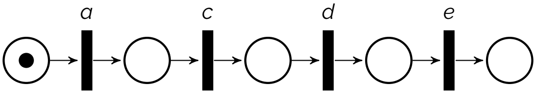
How to compute Quality Criteria

- It exists various metrics for a quality criteria
- Based on process modelling metrics (Petri Net)
- Based on Workflow net
- Based on Process Tree
- Based on alignment algorithms

Examples I

Let us consider the following log

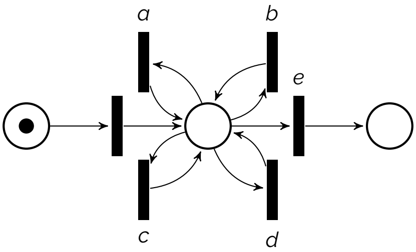
$$L = [\langle a, c, d, e\rangle^{99}, \langle d, a, b, e\rangle^{85}, \langle a, d, c, e\rangle^{56}, \langle a, d, b, e\rangle^{21}, \langle a, b, d, e\rangle^{15}, \langle d, a, c, e\rangle^{6}]$$



- Fitness: bad
- Simplicity: good
- Precision: good
- Generalization: bad

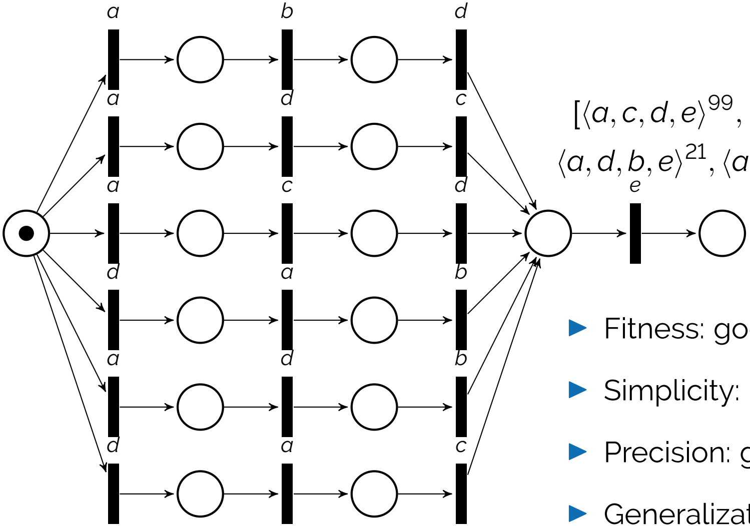
Examples II

$$L = [\langle a, c, d, e\rangle^{99}, \langle d, a, b, e\rangle^{85}, \langle a, d, c, e\rangle^{56}, \langle a, d, b, e\rangle^{21}, \langle a, b, d, e\rangle^{15}, \langle d, a, c, e\rangle^{6}]$$



- Fitness: good
- Simplicity: good
- Precision: bad
- Generalization: good

Examples III



$$[\langle a,c,d,e\rangle^{99}, \langle d,a,b,e\rangle^{85}, \langle a,d,c,e\rangle^{56}, \langle a,d,b,e\rangle^{21}, \langle a,b,d,e\rangle^{15}, \langle d,a,c,e\rangle^{6}] \quad (5)$$

- Fitness: good
- Simplicity: bad
- Precision: good
- Generalization: bad

Outline

Complexity of a Process Model I

Modelling complex business processes is difficult and people make numerous errors. It has been shown in empirical studies that about 20% of models have design flaws...
In Kristian Bisgaard Lassen et al, **Complexity metrics for Workflow nets, Information and Software Technology**, Volume 51, Issue 3, 2009, Pages 610-626, ISSN 0950-5849, they define 3 metrics.

1. **Extend Cardoso metrics** (J. Cardoso Transactions on Enformatika (sixth ed.), Systems Sciences and Engineering, vol. 8, Springer-Verlag, Berlin, Budapest, Hungary (2005), pp. 213-218)
   It is based on the presence of certain splits and joins in the syntactical process definition (based on Weyuker's properties[4]) that give comlexity measure to determine if a program can be categorized as good, structured, and comprehensive.

Complexity of a Process Model II

Cardoso metrics:

- Activity complexity: calculate the number of activities a process has
- Control-flow complexity: based on splits, joins loops and ending
- Data-flow complexity: data complexity and mapping, composed of several sub-metrics (data complexity, interface complexity, and interface integration complexity)
- Resource complexity: based on resources access during activities

The metric is based on the number of subsets of places reachable form a place.

Complexity of a Process Model III

2. **Extend McCabe Cyclomatic metric** (T. McCabe IEEE Transactions on Software Engineering, 2 (1976), pp. 308-320 control flow graph of procedure of a program) well-kown for measuring the control-flow graph of a procedure of a programme.
   The metric is based on the number of edges, vertex and strongly connected components.
3. **Structuredness metric**
   It is based on "behavioral" pattern. It better tries to capture the complexity of the model as it is perceived by humans. It iteratively analyzes the structure of the model and assigns penalties to undesirable constructs from a complexity point of view.

[4] Weyuker, E.J., Evaluating software complexity measures. IEEETransactions on Software Eng., 1988. 14(9): p. 1357-1365

Correspondance between process execution I

In J.E. Cook el al. **Software Process Validation: Quantitatively Measuring the Correspondence of a Process to a Model**. ACM Transactions on Software Engineering and Methodology (TOSEM), 8:147–176, April 1999 Their aim is to measure the level of correspondence between a process execution and a process model.

Their ambition is to answer the questions:

- Does our model reflect what we actually do?
- Do we follow our model?

They define two metrics:

1. Simple String Distance metric
2. Non-linear String Distance metric

Correspondance between process execution II

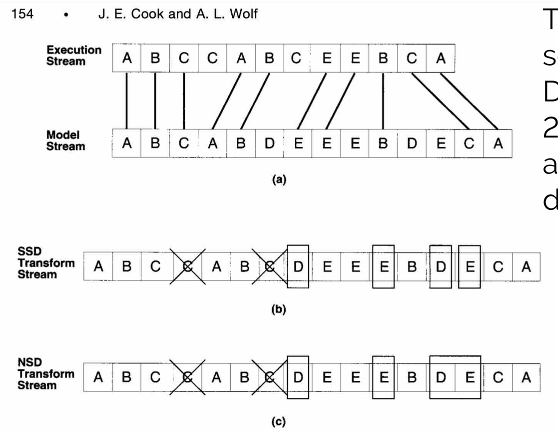


Fig. 3. Example execution and model event streams (a), with execution stream transformed for the SSD metric (b) and the NSD metric (c) calculations.

The metrics are based on comparing sequences. Research on sequencing DNA have been very popular since the 2000s and a lot of sequence alignment algorithms have been developped.


D'ici, on voit + loin !
La Rochelle Université
univ-larochelle.fr

Event logs

R. Champagnat, M. Rabah, M. Trabelsi et al.

2023-2024

Week 36	Introduction
Week 37	Process discovery (α-Algorithm)
Week 38	Metrics and quality of discovered models
Week 39	Raw traces/ modelled traces (case study)
Week 40	Advanced process mining algorithms
Week 41	Advanced process mining algorithms
Week 42	Conformance checking
Week 46	Decision mining in processes
Week 47	Trace clustering
Week 48	Trace profile
Week 49	Case study
Week 50	Case study defense

Outline

Outline

Traces I

- Business Process software
- Software for public services
- Information seeking

Generates a lot of data.

A use trace is a footprint left by a user when using a software.

- At short term, the objective is providing feedback for the production teams.
- At long term, a smart assistant could be designed to help the user to perform some tricky tasks or repetitive actions.

Traces II

Improve quality

- reproducing anomalies situation
- validate user experience
- determine performance criteria

Analysis based on experience rather than knowledge

Difficult to analyse Information Seeking systems

- Task are defined step by step during its realization
- No exact goal
- No indentified means to rich its goal
- Depends on the context (information found during the process)

Outline

1. Introduction
2. Traces
3. Event Log
4. Outlier
5. Sampling Event Log
6. Privacy

Use trace I

Laflaquière, Julien et al. (2006). **Trace-Based Framework for Experience Management and Engineering**.

- A Trace-Based Framework for Experience Management and Engineering is considered as a recording of a computer-mediated activity that is potentially constructed from variety of sources (log-files, video, transcripts, etc)
- Trace lifecycle:
  - Collecting (deciding with what to collect and how)
  - Transformation (automatically ou manually filtering rearranging or adding information)
  - Presentation (visualization and involves choosing what to present and how)
- In a **Trace-Based System** each **trace** must always be associated to an explicit **trace model**

Use trace II

A use trace is a temporal sequences of observed items.

- Order-relation that organizes trace data relatively to a time base
- Observed item indicates that trace data result form an observation

The objective is to deal with use traces that "make sense"

- Qualitative approaches are proposed in ethnographic and ergonomics research
- Quantitative approaches are based on log-files. They are obtained by passive observation and are used to calculate some statistical insights
- Use trace approaches: in between

Use trace III

Trace model

A Trace Model is an ontology $M_T = (C; \leq_C; \leq_R; T; A; \sigma_A; \sigma_R)$ consisting of

- a set of concepts C organized in hierarchy with an order relation $\leq_C$
- a set of relations R organized with $\leq_R$
- a relation signature $R \rightarrow C \times C$
- a set of data types T
- a set of attributes A
- and an attribute signature $A \rightarrow C \times T$.

Use trace IV

Trace

A trace is a quintuplet $(M_T, D_p, O_{tr}, R_t, R_s)$ where

- M_T is the associated trace model;
- D_p is a temporal domain $(T, <)$ with T a set of time instants and $<$ an order on T;
- O_{tr} is a set of objects O, $O_{tr} = O_0, O_1, ..., O_n$ such as $\forall O_i \in O_{tr}, f(O_i) \in C$, with f a labelling function $f : O_{tr} \to C$
- $R_t \subseteq D_p \times D_p \times O_{tr}$ is a relation representing the structural links between objects
- $\forall R_{si} \in R_s, g(R_{si}) \in C$, with g a labelling function $g : R_s \to C$.

Use trace V

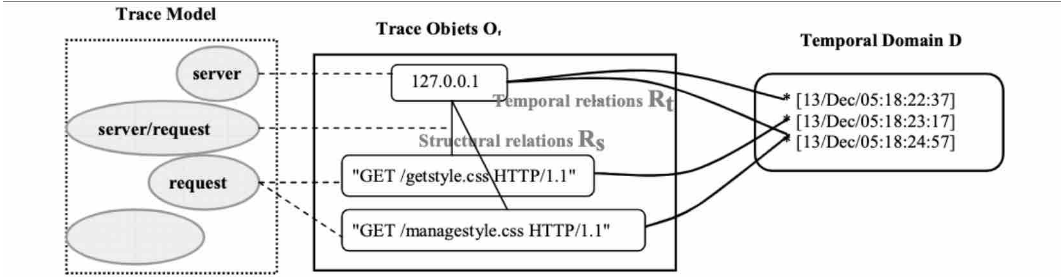


Fig. 3. In this example, the trace model is a set of concepts (server, request, username). Trace objects (one server and two requests) are related to the temporal domain D_p through R_t (note the server is related to a time interval). Traces objects have structural relations through R_s.

Outline

Raw Traces

May be:

- Flat file
- Spreadsheet
- Transaction log
- Database table
- Data warehouse

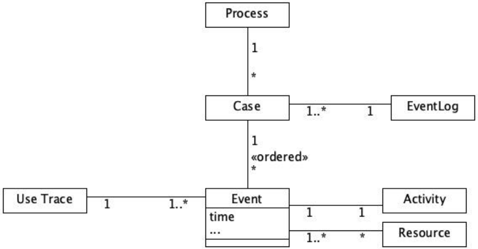
Not always structured and well-described by meta data.

The origin of the raw data could be:

- Web pages
- emails
- PDF documents
- scanned text
- screen scraping

Data need to be extracted and converted into event logs.

Modelled Traces I



- Each event refers to a case, an activity, and a point in time.
- An event log can be seen as a collection of cases.
- A case can be seen as a trace/sequence of events.

Modelled Traces II

Event	time	Process	Case	Activity
e_1	t_1	p_1	c_1	a_1
e_2	t_2	p_1	c_1	a_3
e_3	t_3	p_1	c_2	a_1
e_4	t_4	p_2	c_3	a_2
e_5	t_5	p_1	c_2	a_3
e_6	t_6	p_2	c_3	a_3
e_7	t_7	p_1	c_2	a_4
e_8	t_8	p_1	c_1	a_4
e_9	t_9	p_2	c_4	a_2
e_{10}	t_{10}	p_1	c_5	a_1
e_{11}	t_{11}	p_2	c_3	a_5
e_{12}	t_{12}	p_2	c_4	a_3
e_{13}	t_{13}	p_1	c_6	a_1
e_{14}	t_{14}	p_1	c_5	a_3
e_{15}	t_{15}	p_2	c_4	a_5
e_{16}	t_{16}	p_1	c_6	a_3
e_{17}	t_{17}	p_1	c_6	a_4
e_{18}	t_{18}	p_1	c_5	a_4

- $L = [\langle a_1, a_3, a_4\rangle^4, \langle a_2, a_3, a_5\rangle^2]$
- $E = \{e_1, e_2, ..., e_{18}\}$
- $A = \{a_1, a_2, a_3, a_4\}$
- $P = \{p_1, p_2\}$
- Instance of p_1 : $C_{p_1} = \{c_1, c_2, c_5, c_6\}$
- Instance of p_2 : $C_{p_2} = \{c_3, c_4\}$
- $\langle a_1, a_3, a_4\rangle$ and $\langle a_2, a_3, a_5\rangle$ correspond to variant

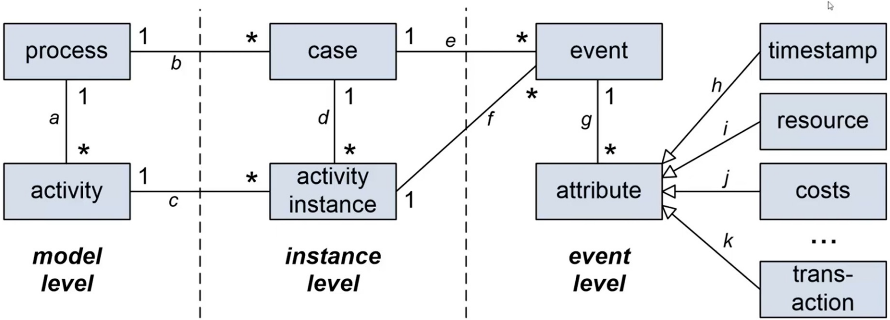
Modelled Traces III

CaseId	User	Timestamp	Activity
1	$user_1$	2016-01-12T10:34:25	home index
1	$user_1$	2016-01-12T10:34:27	home languages
1	$user_1$	2016-01-12T10:34:28	language selection
1	$user_1$	2016-01-12T10:34:31	catalog show
2	$user_2$	2016-01-12T10:34:26	home index
2	$user_2$	2016-01-12T10:34:29	home periods
2	$user_2$	2016-01-12T10:34:30	catalog show

$user_1$ home-index → home-languages → language-selection → catalog-show

$user_2$ home-index → home-periods → catalog-show

More refined view : activity instances



Ambiguity in traces

We can only observed activities that has footprint, but:

Event	time	Process	Case	activity
e_1	t_1	p_1	c_1	start a_1
e_2	t_2	p_1	c_1	start a_1
e_3	t_3	p_1	c_1	complete a_1
e_4	t_4	p_1	c_1	complete a_1

5.2 Event Logs 133

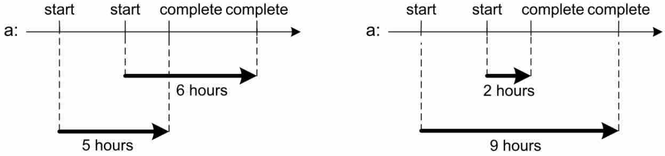


Fig. 5.5 Two scenarios involving two activity instance leaving the same footprint in the log

eXtensible Event Stream I

XES (`www.xes-standard.org`) is a standard for storing and exchanging event logs.

> The XES standard defines a grammar for a tag-based language whose aim is to provide designers of information systems with a unified and extensible methodology for capturing systems behaviors by means of event logs and event streams.

eXtensible Event Stream II

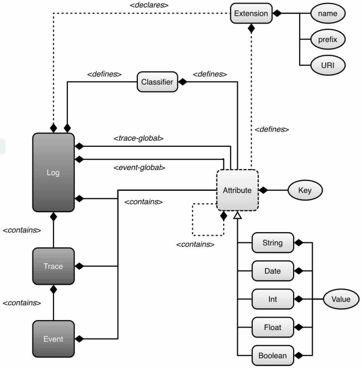


Fig. 5.7 Meta model of XES [64]. A log contains traces and each trace contains events. Logs, traces, and events have attributes. Extensions may define new attributes and a log should declare the extensions used in it. Global attributes are attributes that are declared to be mandatory. Such attributes reside at the trace or event level. Attributes may be nested. Event classifiers are defined for the log and assign a "label" (e.g., activity name) to each event. There may be multiple classifiers

eXtensible Event Stream III

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!-- This file has been generated with the OpenXES library. It conforms -->
<!-- to the XML serialization of the XES standard for log storage and -->
<!-- management. -->
<!-- XES standard version: 1.0 -->
<!-- OpenXES library version: 1.0RC7 -->
<!-- OpenXES is available from http://www.openxes.org/ -->
<log xes.version="1.0" xes.features="nested-attributes" openxes.version="1.0RC7">
        <extension name="Lifecycle" prefix="lifecycle" uri="http://www.xes-standard.org/lifecycle.xesext"/>
        <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext"/>
        <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
        <classifier name="Event␣Name" keys="concept:name"/>
        <classifier name="(Event␣Name␣AND␣Lifecycle␣transition)" keys="concept:name␣lifecycle:transition"/>
        <string key="concept:name" value="XES␣Event␣Log"/>
        <trace>
                <string key="concept:name" value="10|1"/>
                <event> <string key="concept:instance" value="0"/>
                        <string key="lifecycle:transition" value="start"/>
                        <date key="time:timestamp" value="1998-05-06T16:00:57.000+02:00"/>
                        <string key="concept:name" value="request"/><string key="task" value="87"/>
                </event>
                ...
        </trace>
</log>
```

Classifier I

A classifier is a function that maps the attributes of an event onto a label.

For any event $e \in E$ and name $n \in ActivityName$, $\#_n(e)$ is the value of attribute n for event e. And $\underline{e}$ is the name of the event

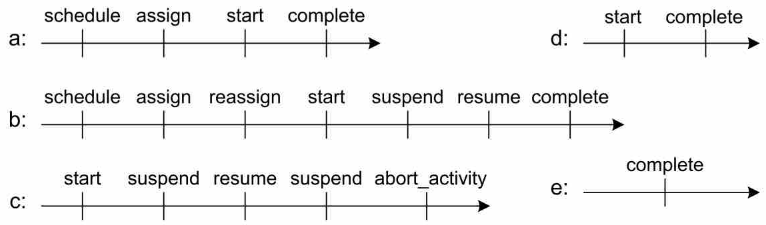


Fig. 5.4 Transactional events for five activity instances

Classifier II

- If events are simply identified by their activity name, then $\underline{e} = \#_{activity}(e)$.
- Instance a in Fig. 5.4 would be mapped onto $\langle a, a, a, a \rangle$.
- In this case α-algortihm would create just one a transition.
- If events are identified by their activity name and transaction type then $\underline{e} = (\#_{activity}(e), \#_{trans}()e)$. Now activity instance a would be mapped onto $\langle (a, schedule), (a, assign), (a, start), (a, complete) \rangle$.

Data Extraction

In the Process Mining book - **Data Science in Action** (Wil M.P. van der Aalst, 2016.) five challenges were highlighted:

- **Event correlation**: how to identify events and their corresponding cases?
- **Timestamps**: when merging data from different sources time may be wrong because of multiple clocks...
- **Snapshot of a longer running process** (missing head or tail)
- **Scoping, knowledge associated to data**
- **Granularity**

Data Quality

- **Missing in log**: activity not recorded
- **Missing in reality**: extra activity recorded
- **Concealed in log**: the activity was recorded and exists but it is hidden in a larger less structured data.
- **Missing attribute**
- **Incorrect attribute**
- **Imprecise attribute**

Guidelines for logging

To create an event log from trace:

1. we need to select the events relevant for the process at hand
2. events need to be correlated to form process instances (cases)
3. events need to be ordered using timestamp information (or have an explicit order)
4. event attributes need to be selected or computed based on the raw data (resource, cost, etc.)

Example

Move on to a real case : how to create an event log from documents ?

-> Database tables extracted from documents

Link to the case presentation [Example on claims documents]

Outline

1. Introduction
2. Traces
3. Event Log
4. Outlier
5. Sampling Event Log
6. Privacy

Outlier I

Ghionna, Lucantonio et al. (2008). **Outlier Detection Techniques for Process Mining Applications**.

Outlier

Exceptional individual trace from a set of traces **or** Infrequent behaviour.

- Important applications in bioinformatics, fraud detection, and intrusion detection, etc.
- Problem in Process Mining: concurrency may produce traces that only differ in the ordering but are not outlier (even if occurs rarely)

Outlier II

Fani Sani et al. (2018). **Applying Sequence Mining for Outlier Detection in Process Mining**.

- Noise versus Outlier.
  - Noise relates to behaviour that does not conform to the process specification or its correct execution.
  - Infrequent behaviour refers to behaviour that is possible according to the process model, but, in exceptional cases of the process.
- The presence of outlier behaviour makes results complex, incomprehensible and even inaccurate.
- Applying filtering on log prior to apply any process discovery algorithm.

Outline

Sampling Event Log I

Fani Sani et al. (2019). **The Impact of Event Log Subset Selection on the Performance of Process Discovery Algorithms**.

Problems:

- Dealing with large event logs
- Meaningful sampling
- Sampling biais

Sampling methods aim to reduce the number of process instances and increase the performance of discovery algorithms

Sampling Event Log II

Subset selection strategies

- **Random Sampling**
- **Biased Sampling Strategies**: first find all variants in an event log and use more advanced strategies (biases) to select them
- **Frequency-based Selection**: This ranking strategy gives higher priority to a variant that has a higher occurrence frequency in the event log
- **Length-based Selection**: sort variants based on their length and choose the longest or the shortest ones first
- **Similarity-based Sampling**: rank variants based on the similarity of them to each other
- **Structure-based Selection**: we consider the presence of unstructured behavior in each variant

Sampling Event Log III

Kabierski, Martin et al. (2018). **How Much Event Data Is Enough? A Statistical Framework for Process Discovery**.

- Statistics for pre-processing event logs (detect unstructured behaviour...)
- Statistics for determining how a newly sampled trace add new information

For instance, with traces $\langle a, d, b, e \rangle$ and $\langle a, b, d, e \rangle$ one can derive the following ordering relations: $a \rightarrow b, a \rightarrow d, b||d, b \rightarrow e, d \rightarrow e$.
Adding the new trace $\langle d, a, b, e \rangle$ changes the deduction on ordering relations as follows: $a \rightarrow b, a||d, b||d, b \rightarrow e, d \rightarrow e$.

Outline

1. Introduction
2. Traces
3. Event Log
4. Outlier
5. Sampling Event Log
6. Privacy

Privacy I

- **Privacy, security, law, and ethics** are key ingredients to protect individuals and organizations from "bad" data science practices.
- Differences between Information **Security** and **Privacy**[1]
  - **Privacy** relates to the idea that the information about individuals or groups that is not advertised to others.
  - **Security** is the practice of preventing unauthorized and malicious access, use, disruption and modification of information.

Privacy referes to the ability to isolate sensitive information.

  - Data should be accurate and stored safely
  - Individuals need to be able to trust the way data are stored and transmitted
  - Not all types of analysis possible are morally defendable.
  - Due to a lack of sufficient data, minority groups may be wrongly classified

Privacy II

- Ensure **privacy** without losing meaningful correlations.
  Hashing can be a powerful tool in the trade-off between privacy and analysis.
- **Privacy** and **anonymization**
  Event logs may contain sensitive or private data. Events refer to actions and properties of customers, employees, etc.
- **Privacy** protection techniques
  - Cryptographic technique
  - Access Control
  - Differential Privacy [2]

Privacy III

- Anonymization techniques
  A study estimated that 87% of the population of the United States can be uniquely identified using the attributes gender, date of birth, and 5-digit zip code[3]. Those three attributes were used to link Massachusetts voter registration records (which includes the name, gender, zip code, and date of birth) to supposedly anonymized medical data from the Group Insurance Commission GIC (which includes gender, zip code, date of birth and diagnosis). The linking between these two tables managed to identify the medical records of the governor of Massachusetts in the medical data[4].
  - K-anonymity
    A table satisfies k-anonymity if every record in the table is indistinguishable from at least k -1 other records with respect to every set of quasi-identifier attributes, such a table is called a k-anonymous table.
    There are many limitations that have been identified for this technique, namely attacks such as homogeneity attack and background knowledge attack.

Privacy IV

- L-Diversity
  Requires each group of quasi-identifier attributes containing at least one representative and distinct sensitive attributes that have equal proportion in order to avoid homogeneity attack and background attack
- t-closeness
  An equivalence class is said to have t-closeness if the distance between the distribution of a sensitive attribute in this class and the distribution of the sensitive attribute in the whole table is no more than a threshold t. A table is said to have t-closeness if all equivalence classes have t-closeness.

Privacy V

- Process mining techniques do not create new data but active use of data and process mining techniques increases the risk of data misuse
- Organizations should continuously balance the benefits of creating and using event data against potential privacy and security problems.

[1] Wang, Tao et al. (2018). Privacy Preservation in Big Data From the Communication Perspective—A Survey. IEEE Communications Surveys & Tutorials.

[2] it seeks at providing rigorous and statistical guarantees against what an adversary can infer and learn over an individual's data. It consists in perturbing the raw records of individuals randomly.

[3] Kunaserkan Kokula Krishna Hari et al. Proceedings of the International Conference on Systems, Science, Control, Communication, Engineering and Technology, ICSSCCET 2015.

[4] Latanya Sweeney. k-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems


D'ici, on voit + loin !
La Rochelle Université
univ-larochelle.fr

Advanced Process Discovery: Inductive Miner, Fuzzy Miner, etc

R. Champagnat, M. Trabelsi et al.

2023-2024

- Week 36 Introduction
- Week 37 Process discovery (α-Algorithm)
- Week 38 Metrics and quality of discovered models
- Week 39 Raw traces/ modelled traces (case study)
- Week 40 Advanced process mining algorithms
- Week 41 Advanced process mining algorithms
- Week 42 Conformance checking
- Week 46 Decision mining in processes
- Week 47 Trace clustering
- Week 48 Trace profile
- Week 49 Case study
- Week 50 Case study defense

Outline

Outline

Representational bias

The representational bias determines the search space and potentially limits the expressiveness of the discovered model.

- Inability to represent concurrency
- Inability to deal with loops
- Inability to represent silent actions
- Inability to represent duplicate actions
- Inability to represent non-free-choice behavior
- Inability to represent hierarchy

Dealing with real log

The real log may contain:

Noise
- Add events in a trace
- Loose events in a trace

Exceptional/infrequent behaviour

Completeness All the variants may not appear in the log, i.e. to discover $a \| b$ we must discover cases containing $\langle ..., a, b, ... \rangle$ and $\langle ..., b, a, ... \rangle$, if a, b, c, d, e are in sequence and in parallel with f, it requires 16 variants to be totally observed.
The assumption that event logs are directly-follows complete is unrealistic for less structured processes and relatively small event logs

Incomplete Case (or trace) may be incomplete (missing the beginning or the end due to data extraction)

Outline

Overview I

Limitation of Process Discovery models:

- Generate models with non-living transitions
- Unable to replay the log

Inductive miner is a family of algorithms that discover a Process Tree model by splitting Log recursively

Inductive miner techniques can deal with:

- infrequent behaviour
- completeness

Overview II

Characteristics:

- Discover "sound" model
- Ability to rediscover the original model
  The property rediscoverability entails that a discovery algorithm is able to discover a model that is language equivalent to the system that underlies the given event log.
- Can deal with huge logs

Inductive mining is currently one of the leading process discovery approaches due to its flexibility, formal guarantees and scalability.

Workflow Net

A Petri net N is a Workflow Net if:

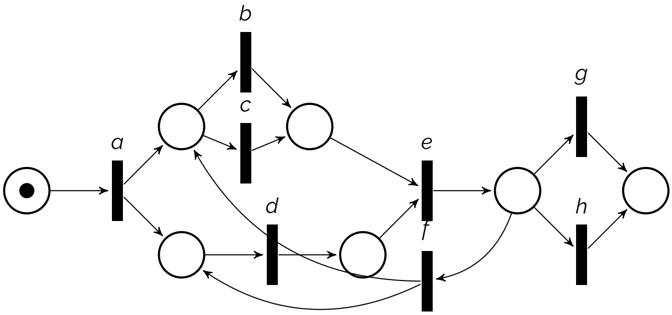
1. Object creation: $\exists\, p_i \in P : \forall t \in T, \nexists Post(t, p_i)$, it contains an input place
2. Object completion: $\exists\, p_o \in P : \forall t \in T, \nexists Pre(p_o, t)$ it contains and output place
3. Connectedness: adding transition $\bar{t}$ from p_o to p_i, then we have $\bar{N} = (P, T \cup \bar{t}, \mathcal{F} \cup \{(p_o, \bar{t}), (\bar{t}, p_i))\}$ is strongly connected.

Sound I

A Workflow Net is sound iff:

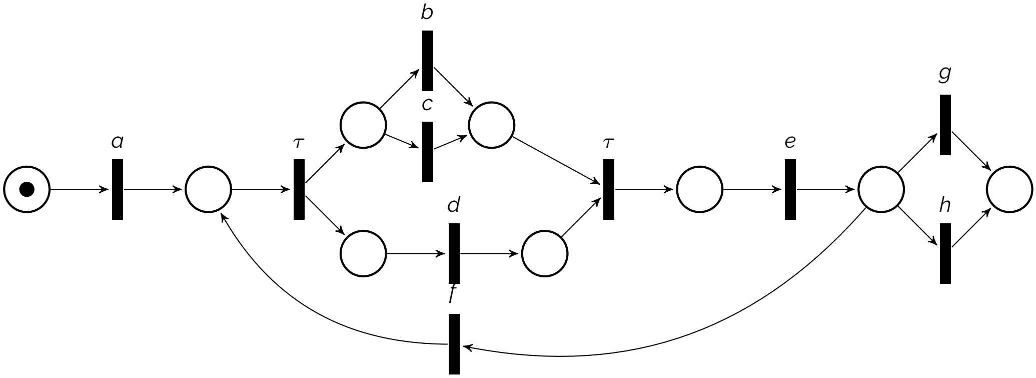
- the net is safe (places cannot hold multiple tokens at the same time);
- proper completion: if the sink place is marked, all other places are empty. $\forall_S([i] \xrightarrow{*} s) \Rightarrow (s \xrightarrow{*} [o])$, for each reachable marking from the input place there exists a sequence of firing that leads to the final marking with $[i]$ the initial marking meaning only p_i holds a token and $[o]$ the final marking meaning only p_o holds a token;
- option to complete: it is always possible to reach the marking that marks just the sink place. $\forall M([i] \xrightarrow{*} s \wedge s \geq [o]) \Rightarrow (s = [o])$;
- Absence of dead parts.

Sound II



Block-Structured Workflow Nets

Block-Structured Workflow Nets is a hierarchical workflow net that can be divided recursively into parts having single entry and exit points.



Process Tree I

A process tree is a compact abstract representation of a block-structured workflow net: a rooted tree in which leaves are labeled with activities and all other nodes are labeled with operators.
A Process Tree is formally defined recursively by:
Let a finite alphabet Σ of activities and a set $\bigoplus$ of operators. Symbole $\tau \notin \Sigma$ denotes the silent activity.

Process Tree

- a, with $a \in \Sigma \cup \tau$, is a Process Tree;
- Let $M_1 ... M_n$ with $n > 0$ be Process Tree and let $\bigoplus$ a Process Tree Operator, then $\bigoplus(M_1, ..., M_n)$ is a Process Tree.

Process Tree II

- Operator $\times$ means the exclusive choice,
- $\rightarrow$ means the sequential execution,
- $\wedge$ means a parallel (interleaved) execution and
- $\circlearrowleft$ a structured loop (with do and redo).

Example: $\rightarrow(a, \circlearrowleft(\rightarrow(\wedge(\times(b,c),d),e),f),\times(g,h))$ correspond to the Block-Structured Workflow net given slide 13.

Convert Process Tree to Workflow Net

Any process tree can be converted to an equivalent WF-net (and BPMN model, etc.) directly by:

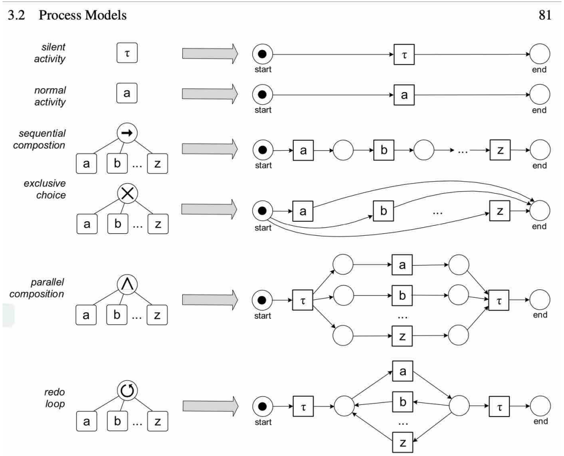


Fig. 3.18 Mapping process trees onto WF-nets

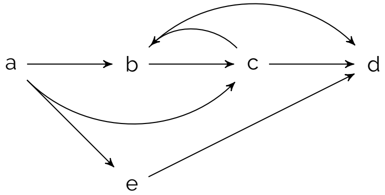
Directly-Follows Graph I

The basic Inductive Miner algorithm uses the Directly-Follows graph that corresponds to the "directly follows" relation ($>_L$) used by the α-algorithm.
It is formally defined by:
Let L be an event log. The Directly-Follows graph of L is $G(L) = (A_L, \mapsto_L, A_L^{start}, A_L^{end})$ with:

- $A_L = \{a \in \sigma | \sigma \in L]\}$ is the set of activities in L
- $\mapsto_L = \{(a, b) \in A \times A | a >_L b\}$ is the directly follows relation
- $A_L^{start} = \{a \in A | \exists_{\sigma \in L} a = first(\sigma)\}$ is the set of start activities
- $A_L^{end} = \{a \in A | \exists_{\sigma \in L} a = last(\sigma)\}$ is the set of end activities

Directly-Follows Graph II

From $L = [\langle a,b,c,d\rangle^3, \langle a,c,b,d\rangle^2, \langle a,e,d\rangle]$



Eventually-Follows Graph I

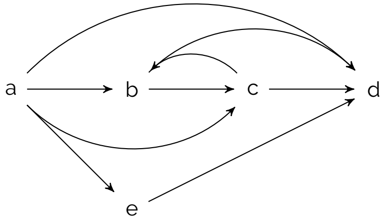
The Eventually-Follows graph corresponds to the relation a is eventually followed by b if there is a trace in the event log in which a happens somewhere before b.

Let L be an event log. The Eventually-Follows graph of L is $G_e(L) = (A_L, \mapsto_L^+, A_L^{start}, A_L^{end})$ with:

- $A_L = \{a \in \sigma | \sigma \in L]\}$ is the set of activities in L
- $\mapsto_L^+$ is the eventually follows relation.
  if there is a non-empty path from a to b in $G(L)$, i.e., there exists a sequence of activities $a_1, a_2, ..., a_k$ such that $k \geq 2$, $a_1 = a$ and $a_k = b$ and $a_i \mapsto_L a_{i+1}$.
  $a \not\mapsto_L^+ b$ if there is no path from a to b in the directly-follows graph.
- $A_L^{start} = \{a \in A | \exists_{\sigma \in L} a = \mathit{first}(\sigma)\}$ is the set of start activities
- $A_L^{end} = \{a \in A | \exists_{\sigma \in L} a = \mathit{last}(\sigma)\}$ is the set of end activities

Eventually-Follows Graph II

From $L = [\langle a,b,c,d\rangle^3, \langle a,c,b,d\rangle^2, \langle a,e,d\rangle]$



Algorithm I

Leemans, Sander & Fahland, Dirk & Aalst, Wil. (2013). Discovering Block-Structured Process Models from Event Logs - A Constructive Approach.

Inductive Miner

1. The Inductive Miner algorithm iteratively splits the initial event log into smaller sublogs using cuts.
2. For any sublog L we can create a directly-follows graph $G(L)$
3. Sublogs will be mined recursively until a sublog will contain just a single activity

Algorithm II

We consider the following cuts:

- exclusive-choice
- sequence
- parallel
- redo-loop

Corresponding to the four Process Tree operators: $\times, \rightarrow, \wedge, \circlearrowleft$

224 7 Advanced Process Discovery Techniques

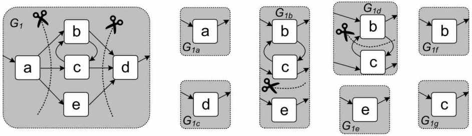
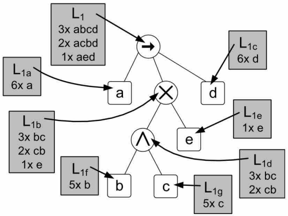


Fig. 7.21 G_1 is the directly-follows graph for $L_1 = [\langle a,b,c,d\rangle^3, \langle a,c,b,d\rangle^2, \langle a,e,d\rangle]$. The event log is recursively cut into smaller sublogs using the directly-follows graphs of these sublogs

Fig. 7.22 The different sublogs created when learning process tree $Q_1 = \rightarrow(a, \times(\wedge(b,c),e),d)$ for $L_1 = [\langle a,b,c,d\rangle^3, \langle a,c,b,d\rangle^2, \langle a,e,d\rangle]$



Cuts I

Let L be an event log with corresponding directly-follows graph $G(L) = (A_L, \mapsto_L, A_L^{start}, A_L^{end})$.
Let $n \geq 1$.
A n-ary cut of $G(L)$ is partition of A_L into pairwise disjoint sets $A_1, A_2, \ldots, A_n : A_L = \bigcup_{i \in 1,\ldots,n} A_i$ and $A_i \cap A_j = \varnothing$ for $i \neq j$.
Notation is $(\oplus, A_1, A_2, \ldots, A_n)$ with $\oplus \in \{\rightarrow, \times, \wedge, \circlearrowleft\}$.
For each type of operator $\rightarrow, \times, \wedge, \circlearrowleft$ sepcific conditions apply:

Exclusive-choice cut (no crossing edges) of $G(L)$ is a cut $(\times, A_1, A_2, \ldots, A_n)$ such that $\forall_{i,j \in [1,\ldots,n]} \forall_{a \in A_i} \forall_{b \in A_j} i \neq j \Rightarrow a \not\mapsto_L b$

Sequence cut (edges crossing one-way only) of $G(L)$ is a cut $(\rightarrow, A_1, A_2, \ldots, A_n)$ such that $\forall_{i,j \in [1,\ldots,n]} \forall_{a \in A_i} \forall_{b \in A_j} i < j \Rightarrow (a \mapsto_L^+ b \wedge b \not\mapsto_L^+ a)$

Cuts II

Parallel cut (all possible crossing edges) of $G(L)$ is a cut $(\wedge, A_1, A_2, ..., A_n)$ such that

- $\forall_{i \in [1,...,n]} A_i \cap A_L^{start} \neq \emptyset \wedge A_i \cap A_L^{end} \neq \emptyset$ and
- $\forall_{i,j \in [1,...,n]} \forall_{a \in A_i} \forall_{b \in A_j} i \neq j \Rightarrow a \mapsto_L b$

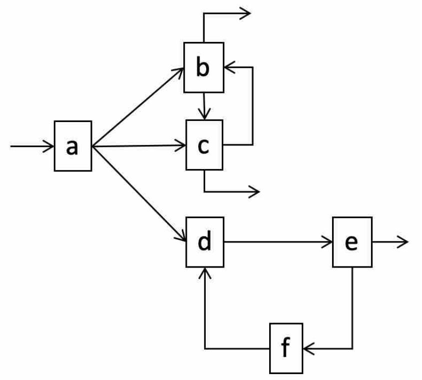
Cuts III

redo-loop cut (identify body and loopback parts; assumption: start/end activities disjoint) of $G(L)$ is a cut $(\circlearrowleft, A_1, A_2, ..., A_n)$ such that

- $n \geq 2$
- $A_L^{start} \cup A_L^{end} \subseteq A_1$
- $\{a \in A_1 | \exists_{i \in [2,...,n]} \exists_{b \in A_i} a \mapsto_L b\} \subseteq A_L^{end}$
- $\{a \in A_1 | \exists_{i \in [2,...,n]} \exists_{b \in A_i} b \mapsto_L a\} \subseteq A_L^{start}$
- $\forall_{i,j \in [2,...,n]} \forall_{a \in A_i} \forall_{b \in A_j} i \neq j \Rightarrow a \not\mapsto_L b$
- $\forall_{i \in [2,...,n]} \forall_{b \in A_i} \exists_{a \in A_L^{end}} a \mapsto_L b \Rightarrow \forall_{a' \in A_L^{end}} a' \mapsto b$
- $\forall_{i \in [2,...,n]} \forall_{b \in A_i} \exists_{a \in A_L^{end}} b \mapsto_L a \Rightarrow \forall_{a' \in A_L^{start}} b \mapsto a'$

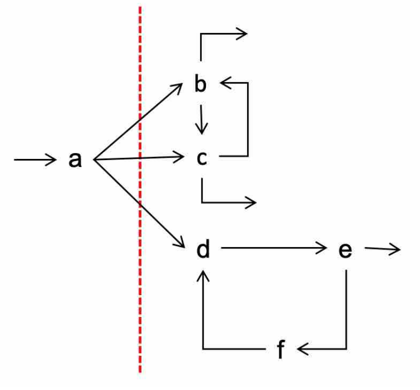
Example I

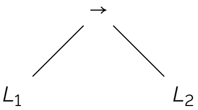
Let us consider the following log: $L = [\langle a, b, c\rangle, \langle a, c, b\rangle, \langle a, d, e\rangle, \langle a, d, e, f, d, e\rangle]$
We derive the following directly-follows graph:



Example II

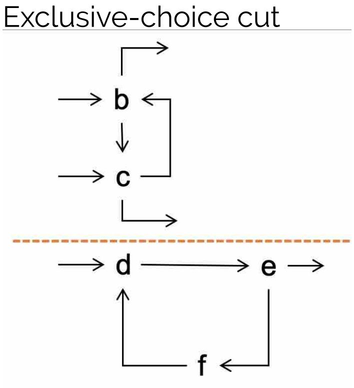
Sequence cut

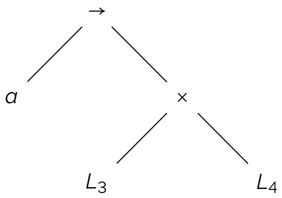




with $L_1 = [\langle a \rangle, \langle a \rangle, \langle a \rangle, \langle a \rangle]$ and
$L_2 = [\langle b, c \rangle, \langle c, b \rangle, \langle d, e \rangle, \langle d, e, f, d, e \rangle]$

Example III

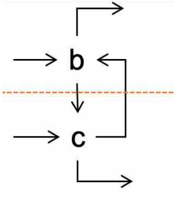


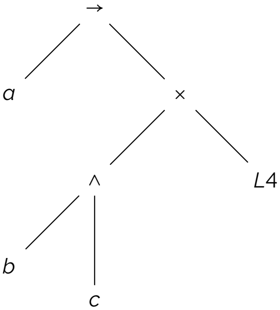


with $L_3 = [\langle b, c\rangle, \langle c, b\rangle]$ and $L_4 = [\langle d, e\rangle, \langle d, e, f, d, e\rangle]$

Example IV

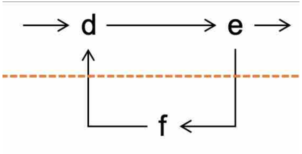
Parallel cut

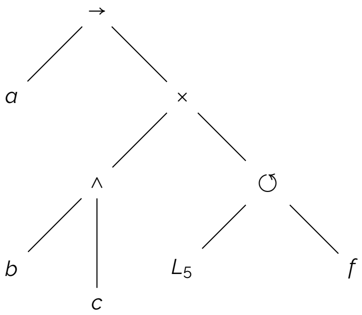




Example V

Redo-loop cut

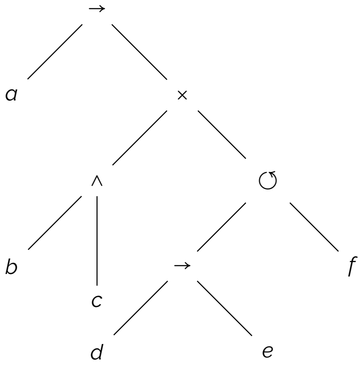




with $L_5 = [\langle d, e \rangle]$

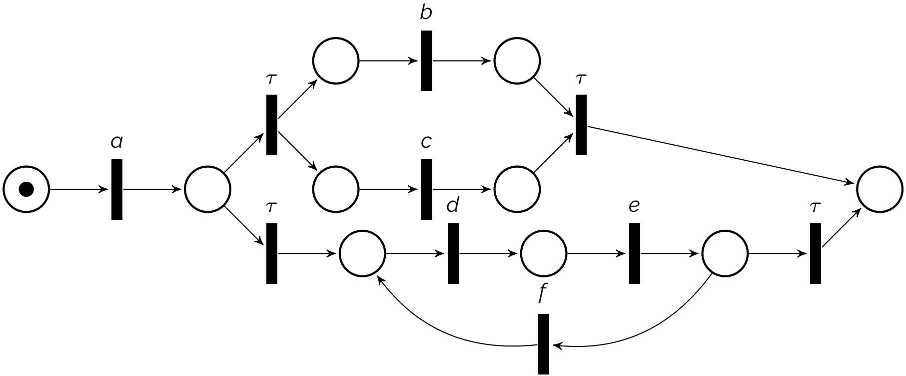
The complete Process Tree is then:

Example VI



Example VII

And the corresponding Workflow net:



Empty traces

Silent activities are only introduced for base cases and empty traces.

- If the sublog is of the form $L' = \left[\langle\rangle^k, \langle a\rangle^l\right]$ with $k, l \geq 1$, then $IM(L') = \times(a, \tau)$ because a is sometimes skipped.
- If a is executed at least once in each trace in the sublog and sometimes multiple times (e.g., $L' = \left[\langle a\rangle^9, \langle a,a\rangle^2, \langle a,a,a\rangle\right]$), then $IM(L') = \circlearrowleft(a, \tau)$.
- In all other cases e.g., $L' = \left[\langle\rangle^3, \langle a\rangle^4, \langle a,a,a\rangle\right]$, $IM(L') = \circlearrowleft(\tau, a)$ because a is executed zero or more times in the traces of sublog L.

Inductive Miner Infrequent I

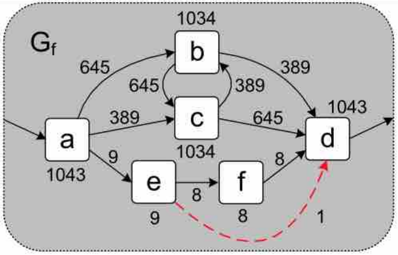
Leemans, Sander & Fahland, Dirk & Aalst, Wil. (2014). Discovering Block-Structured Process Models from Event Logs Containing Infrequent Behaviour.

Deals with variants with low frequency.

Let us consider the log $L = \left[\langle a,b,c,d\rangle^{645}, \langle a,c,b,d\rangle^{389}, \langle a,e,f,d\rangle^{8}, \langle a,e,d\rangle\right]$.
Variant $\langle a,e,d\rangle$ is infrequent.

Inductive Miner Infrequent II

The corresponding Directly-Follows graph is given by:



Where the numbers indicate frequencies, e.g., activity *b* was executed 1034 times and was directly followed by activity *c* 645 times.
The basic idea is to:

- filter edge with low frequency ($e \mapsto d$)

Inductive Miner Infrequent III

- filter activity with low frequency (*e* and *d*). The log is then $L' = \left[\langle a,b,c,d\rangle^{645}, \langle a,c,b,d\rangle^{389}, \langle a,d\rangle^{8}, \langle a,d\rangle\right]$
- filter edge with low frequency from the Eventuall-Follows graph ($\mapsto_L$)

The filtering can also be applied to log splitting (adapting the cut operators). For instance (with $\Sigma_1 = \{a\}$, $\Sigma_2 = \{b\}$):

- Behaviour that violates the $\times$ operator is the presence of activities from more than one subtree in a single trace. For instance, the trace $t_1 = \langle a,a,a,a,b,a,a,a,a\rangle$ contains activities from both Σ_1 and Σ_2. Σ_1 explains the most activities, is most frequent. All activities not from Σ_1 are considered infrequent and are discarded: $\langle a,a,a,a,a,a,a,a\rangle \in L_1$. In t_2, the split $\langle a,a,a,a\rangle \in L_1$, $\langle b,b,b,b,b\rangle \in L_2$ discards the least events.

Inductive Miner Infrequent IV

- Behaviour that violates the $\rightarrow$ operator is the presence of events out of order according to the subtrees. For instance, in the trace $t_2 = \langle a, a, a, a, b, b, b, b, a, b \rangle$, the last a occurs after a b, which violates the $\rightarrow$. Filtering infrequent behaviour is an optimization problem: the trace is to be split in a least-events-removing way.

Inductive Miner Incompleteness

Leemans, Sander & Fahland, Dirk & Aalst, Wil. (2014). Discovering Block-Structured Process Models from Incomplete Event Logs.

Tackle the issue of missing behavior due to the incompleteness of the event log

The IMC algorithm uses so-called “probabilistic activity relations” based on both the directly-follows graph and the eventually-follows graph. These are used to select the “most likely cut” even if the requirements are not fully satisfied

Inductive Miner Directly-Follows based

- Apply Inductive Miner techniques on the directly-follows graph directly without creating sublogs.
- Directly-follows graphs can be computed in a single pass over the event log, and their computation can even be parallelized, for instance using highly-scalable map-reduce techniques
- Pros: extremely scalable

Week 36	Introduction
Week 37	Process discovery (α-Algorithm)
Week 38	Metrics and quality of discovered models
Week 39	Raw traces/ modelled traces (case study)
Week 40	Advanced process mining algorithms
Week 41	Advanced process mining algorithms
Week 42	Conformance checking
Week 46	Decision mining in processes
Week 47	Trace clustering
Week 48	Trace profile
Week 49	Case study
Week 50	Case study defense

Outline

1. Introduction
2. Inductive Miner
3. Fuzzy Miner
4. Other Algorithms
5. Conclusion

Overview I

Günther et al. (2007). Fuzzy Mining – **Adaptive Process Simplification Based on Multi-perspective Metrics**.

- Deal with unstructured processes that generally generate spaghetti-like model.
- Demo with disco

Overview II

- The **fuzzy miner** was developed to simplify the mined process model.
- The problem was that the resulting model tends to show all details without providing an abstraction. Where in reality, **activities** and **relations** can be clustered or removed depending on their role in the process.

Adaptative approach for process simplification inspired by the route map.

- works similarly to a **GPS** software. It tries to discover models depending to user desires.
- If the user **zooms** in, the model will include more details. When the user **zooms** out, the model is clustered and becomes **fuzzier** (which gives the algorithm its name).

Overview III

Fig. 2. Example of a road map.

Overview IV

Concepts:

Aggregation To limit the number of information items displayed, maps often show coherent clusters of low-level detail information in an aggregated manner.

Abstraction Lower-level information that is insignificant in the chosen context is simply omitted from the visualization.

Emphasis More significant information is highlighted by visual means such as color, contrast, saturation, and size.

Customization Maps are specialized in a defined local context, have a specific level of detail, and a dedicated purpose

Overview V

Based on **metrics**:

Significance

- Measures the relative **importance** of each activity
- One example for measuring significance is by **frequency**, i.e. events or precedence relations which are observed more frequently are deemed more significant

Correlation

- Measures how closely related two events following one another are.

Overview VI

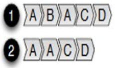
Sketch of approach for process simplification:

- **Highly significant behaviour is preserved**, i.e. contained in the simplified model.
- **Less significant but highly correlated behaviour** is aggregated, i.e. hidden in clusters within the simplified model.
- **Less significant and lowly correlated behaviour** is abstracted from, i.e. removed from the simplified model.

Log-Based Process Metrics I

Three Primary Types of Metrics

Unary Significance describes the relative importance of an event class



Event class	A	B	C	D
Frequency	4	1	2	2
Significance	1.0	0.25	0.5	0.5

- **Unary Frequency Significance**: The more often a certain event class was observed in the log, the more significant it is.
- **Unary Routing Significance**: The higher the number and significance of predecessors for a node (i.e., its incoming arcs) differs from the number and significance of its successors (i.e., outgoing arcs), the more important that node is for routing in the process.

Log-Based Process Metrics II

Binary Significance describes the relative importance of a precedence relation between two event classes, i.e. an edge in the process model.

Prec. Relation	A→B	B→A	A→C	C→D	A→A
Frequency	1	1	2	2	1
Significance	0.5	0.5	1	1	0.5

- **Binary Frequency Significance**: The more often two event classes are observed after one another, the more significant their precedence relation.
- **Binary Distance Significance**: The more the significance of a relation differs from its source and target nodes' significances, the less its distance significance value

Log-Based Process Metrics III

Binary correlation Measures the distance of events in a precedence relation, i.e. how closely related two events following one another are (need timestamp).

Prec. Relation	A→B	B→A	A→C	C→D	A→A
⌀ Duration	3	1	2	3	1
Correlation	0.33	1.0	0.5	0.33	1.0

- **Proximity Correlation**: Event classes that occur shortly after one another are highly correlated.

Log-Based Process Metrics IV

- **Originator Correlation**: The correlation between event classes is determined from the names of the persons, who have triggered two subsequent events. The more similar these names, the higher correlated the respective event classes.
- **Endpoint Correlation**: More similar activity names of subsequent events will be interpreted as higher correlation.
- **Data Type Correlation**: Event classes are highly correlated if subsequent events share a large amount of data types.
- **Data Value Correlation**: Event classes are highly correlated if subsequent events share a large amount of data values.

Adaptive Graph I

Process Model

- All **event classes** found in the log are translated to **activity nodes**, whose importance is expressed by **unary significance**.
- For every observed **precedence relation** between **event classes**, a corresponding directed edge is added to the process model. This **edge** is described by the **binary significance and correlation** of the ordering relation it represents.

Adaptive Graph II

Three transformation methods are applied to simplify the model

Conflict Resolution Whenever two nodes in the initial process model are connected by edges in both directions, they are defined to be in conflict.
Possible situations:

- **Length-2-loop**: after executing A and B in sequence, one may return to A and start over. The conflicting ordering relations between these activities are explicitly allowed in the original process, and thus need to be preserved.
- **Exception**: The process orders $A \rightarrow B$ in sequence, however, during real-life execution the exceptional case of $B \rightarrow A$ also occurs. The "weaker" relation needs to be discarded to focus on the main behaviour.

Adaptive Graph III

- **Concurrency**: *A* and *B* can be executed in any order. Both conflicting ordering relations need to be removed from the process model.

Edge Filtering isolates the most important behaviour by removing the globally least significant edges, leaving only highly significant behavior. It uses a weighted sum of its significance and correlation.

Adaptive Graph IV

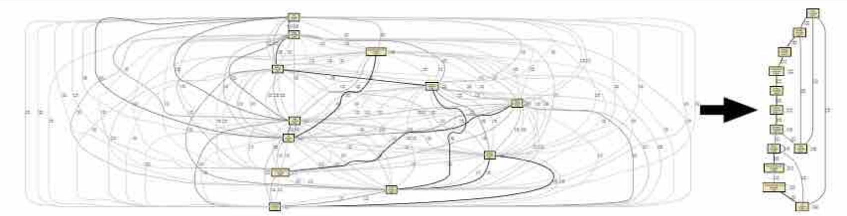


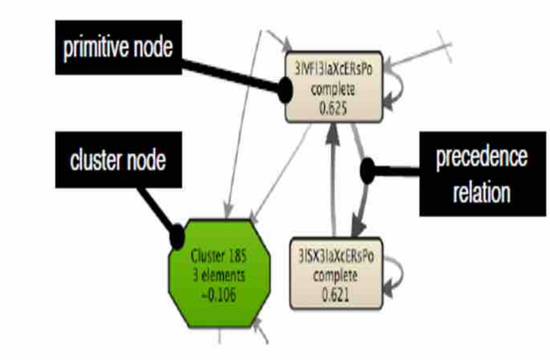
Fig. 6. Example of a process model before (left) and after (right) edge filtering.

Node Aggregation and Abstraction Preserves highly correlated groups of less-significant nodes as aggregated clusters, while removing isolated, less-significant nodes.

- First phase: every node whose unary significance is below a threshold will either be aggregated or abstracted from.

Adaptive Graph V

- Second phase: merge the clusters



Detail

Günther, CW 2009, "**Process mining in flexible environments**"`https://doi.org/10.6100/IR644335`

Detail

Its purpose is to answer the question: “How important is the behavior explicitly shown in the model, compared to behavior that has been aggregated or abstracted from?”

Let F be a fuzzy model. Let N be the set of all primitive nodes in F , i.e., nodes that are explicit, aggregated, or abstracted from. Let $E \subseteq N$ be the subset of all explicit nodes in F. Further, let $s \rightarrow \mathbb{R}_0^+$ be a function that assigns to each node in F its unary significance. The detail dt of a fuzzy model F is defined as

$$dt = \frac{\sum_{e \in E} s(e)}{\sum_{n \in N} s(n)} \tag{1}$$

Conformance

Conformance

Measures the alignment between a fuzzy model and an event log.

The behavior recorded in each trace of the log is replayed in the fuzzy model. Any event in the log that is not valid in the given fuzzy model given the previous execution history, counts as a deviation.

Let L be an event log, and F be a fuzzy model. Let d be the number of deviations, i.e., the number of events in L that cannot be explained by F. The conformance C between F and L is defined as:

$$C = \frac{M(L) - d + 1}{M(L) + 1} \tag{2}$$

Outline

α-algorithm extensions I

Medeiros et al. (2004). **Process Mining for Ubiquitous Mobile Systems: An Overview and a Concrete Algorithm**.

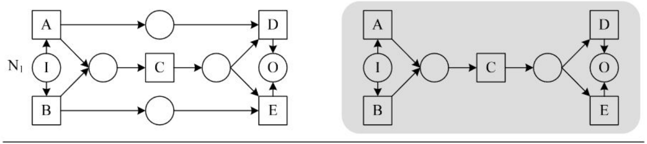
α^{+}

Deal with short-loops

Wen et al (2007). **Mining process models with non-free-choice Constructs**.

α^{++}

Deal with non-free choice



Flexible Heuristic Miner I

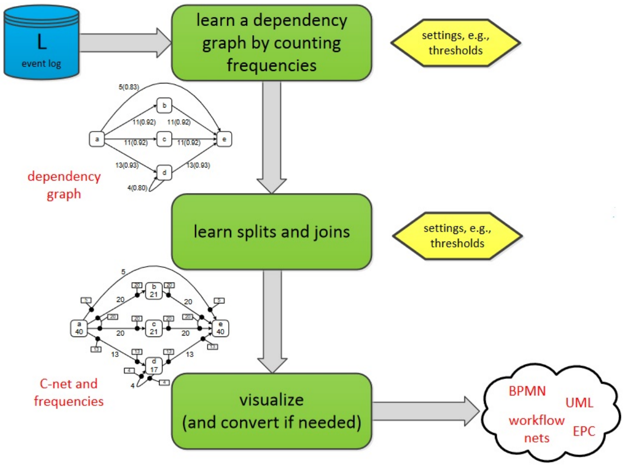
Weijters et al (2011). Flexible Heuristics Miner (FHM). Motivation

- Low-structured processes
- Noise

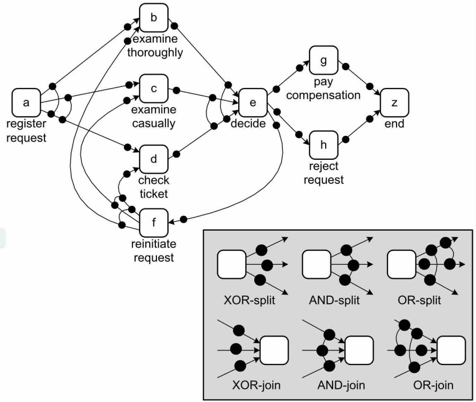
Take **frequencies** of events and sequences into account when constructing a process model.
Based on:

- **Causal Nets**
- **Dependency measures**
  A frequency-based metric is used to indicate how certain we are that there is a truly dependency relation between two events *a* and *b* (consider direct successor and length-two loops).

Flexible Heuristic Miner II



Flexible Heuristic Miner III



Causal nets **Fig. 3.12** Causal net C_1

Flexible Heuristic Miner IV

76 3 Process Modeling and Analysis

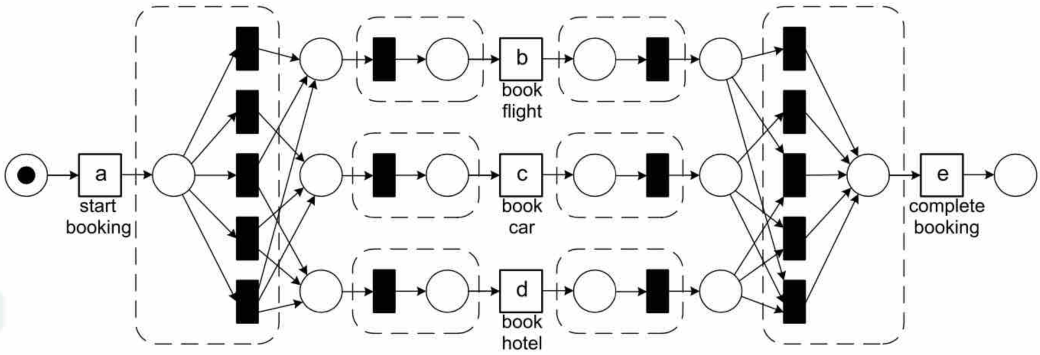


Fig. 3.14 A C-net transformed into a WF-net with silent transitions: every "sound run" of the WF-net corresponds to a valid sequence of the C-net C_2 shown in Fig. 3.13

Heuristic Miner : Learning dependency graph

- Reminder : *Direct succession* : $a_1 > a_2$ if there is a trace such that a_1 is immediately followed by a_2 in a log;
- From $L = [\langle a,e\rangle^5, \langle a,b,c,e\rangle^{10}, \langle a,c,b,e\rangle^{10}, \langle a,b,e\rangle, \langle a,c,e\rangle,$
$\langle a,d,e\rangle^{10}, \langle a,d,d,e\rangle^2, \langle a,d,d,d,e\rangle]$ count all the direct succession in L.

|> L|	a	b	c	d	e
a	0	11	11	13	5
b	0	0	10	0	11
c	0	10	0	0	11
d	0	0	0	4	13
e	0	0	0	0	0

Heuristic Miner : Learning dependency graph

Direct succession Measure

Let L be an event log over A and a, b A . |a >L b| is the number of times a is directly followed by b in L, i.e.,

$$|a >_L b| = \sum_{\sigma \in L} L(\sigma) \times |\{1 \leq i < |\sigma| \mid \sigma(i) = a \wedge \sigma(i+1) = b\}|$$

Dependency Value

$$|a \Rightarrow_L b|$$

is the value of the dependency relation between a and b:

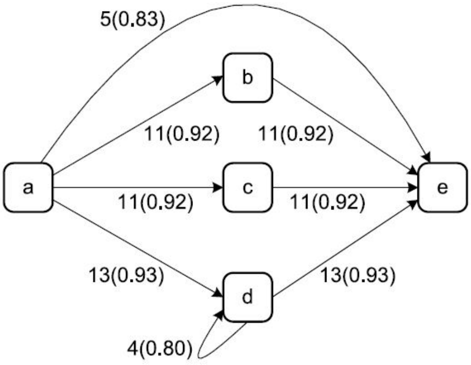
$$|a \Rightarrow_L b| = \begin{cases} \frac{|a >_L b| - |b >_L a|}{|a >_L b| + |b >_L a| + 1} & \text{if } a \neq b \\ \frac{|a >_L a|}{|a >_L a| + 1} & \text{if } a = b \end{cases}$$

Heuristic Miner : Learning dependency graph

$\vert\Rightarrow_L\vert$	a	b	c	d	e
a	$\frac{0}{0+1}=0$	$\frac{11-0}{11+0+1}=0.92$	$\frac{11-0}{11+0+1}=0.92$	$\frac{13-0}{13+0+1}=0.93$	$\frac{5-0}{5+0+1}=0.83$
b	$\frac{0-11}{0+11+1}=-0.92$	$\frac{0}{0+1}=0$	$\frac{10-10}{10+10+1}=0$	$\frac{0-0}{0+0+1}=0$	$\frac{11-0}{11+0+1}=0.92$
c	$\frac{0-11}{0+11+1}=-0.92$	$\frac{10-10}{10+10+1}=0$	$\frac{0}{0+1}=0$	$\frac{0-0}{0+0+1}=0$	$\frac{11-0}{11+0+1}=0.92$
d	$\frac{0-13}{0+13+1}=-0.93$	$\frac{0-0}{0+0+1}=0$	$\frac{0-0}{0+0+1}=0$	$\frac{4}{4+1}=0.80$	$\frac{13-0}{13+0+1}=0.93$
e	$\frac{0-5}{0+5+1}=-0.83$	$\frac{0-11}{0+11+1}=-0.92$	$\frac{0-11}{0+11+1}=-0.92$	$\frac{0-13}{0+13+1}=-0.93$	$\frac{0}{0+1}=0$

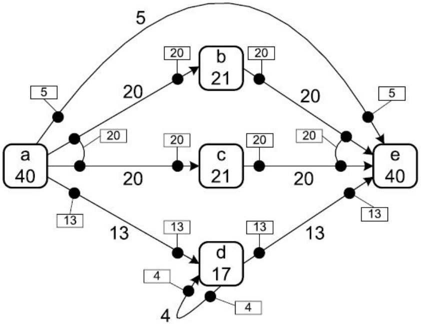
Heuristic Miner : Learning dependency graph

- Dependency graph using a threshold of 2 for $|>_L|$ and 0.7 for $|\Rightarrow_L|$: each arc shows the $|>_L|$ value and the $|\Rightarrow_L|$ value between brackets. For example, $|a >_L d| = 13$ and $|a \Rightarrow_L d| = 0.93$



Heuristic Miner : Learning Splits and Joins

- C-net derived from *L*. Each node shows the frequency of the corresponding activity. Every arc has a frequency showing how often both activities agreed on a common **binding**. The frequencies of input and output bindings are also depicted, e.g., 20 of the 40 occurrences of *a* were followed by the concurrent execution of *b* and *c*



Genetic Miner I

Medeiros et al. (2007). **Genetic process mining: An experimental evaluation**. Data Mining and Knowledge Discovery.

Motivation:

- non-free choice (synchronization and choice)
- invisible tasks
- duplicate tasks
- Noise

Try to mimic the process of evolution. Such approaches are not deterministic and depend on randomization to find new alternatives.

Genetic Miner II

Principle:

- **Initialization**
  A first generation of individuals is created. An individual is a process model. Using the activity names appearing in the log, process models are created randomly.
- **Selection**
  The fitness of each individual is computed. A fitness function determines the quality of the individual in relation to the log
- **Reproduction**
  Populations evolve by selecting the fittest individuals and generating new individuals using genetic operators such as crossover (combining parts of two or more individuals) and mutation (random modification of an individual)

Genetic Miner III

- **Termination**
  The best individuals move on to the next round (elitism) or are used to produce new offspring
  The evolution process terminates when a satisfactory solution is found, i.e., a model having at least the desired fitness

Difficulties:

- Define the internal representation (the search space of a genetic algorithm) by a causal matrix (expresses the task dependencies)
  A Causal Matrix is a tuple $CM = (A, C, I, O)$, where:
  - A is a finite set of activities
  - $C \subseteq A \times A$ is the causality relation
  - $I : A \rightarrow P(P(A))$ is the input condition function
  - $O : A \rightarrow P(P(A))$ is the output condition function

  Such that

Genetic Miner IV

- $C = \{(a_1, a_2) \in A \times A | a_1 \in \bigcup I(a_2)\}$
- $C = \{(a_1, a_2) \in A \times A | a_2 \in \bigcup O(a_1)\}$
- $C \bigcup \left\{(a_o, a_i) \in A \times A | a_o \bullet^C = 0 \wedge \bullet a_i^C = 0\right\}$ is a strongly connected graph

- Define the fitness measure
- Genetic operators
  They should ensure that all points in the search space defined by the internal representation may be reached when the genetic algorithm runs

Two Step Approach I

Aalst et al. (2010). **Process mining: A two-step approach to balance between underfitting and overfitting.** Software and Systems Modeling. 9. 87-111.

Motivation: enable the user to control the balance between "overfitting" and "underfitting" and discover concurrency

Approach

1. Construct a transition system based on prefix, on postfix or on prefix and on postfix

# Instances	Log Traces
1435	A B D E A
946	A C D G H F A
764	A C G D H F A
54	A C G H D F A
1	A C D G G H F A

(a) Log L_1

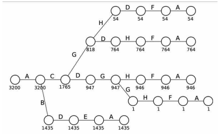


Fig. 2. Prefix Automaton

Two Step Approach II

2. A Petri Net is synthesized from the transition system resulting (state-based regions or language-based regions)

Used to mine the objections handled by the Municipality of Heusden.

Outline

Which Process Discovery algorithm is the best? I

Rozinat el al. (2007). **Towards an evaluation framework for process mining algorithms. Reactivity of Solids.**

With the following log: $L = [\langle A, B, D, E, I \rangle, \langle A, C, D, G, H, F, I \rangle, \langle A, C, G, D, H, F, I \rangle, \langle A, C, H, D, F, I \rangle, \langle A, C, D, H, F, I \rangle]$

Which Process Discovery algorithm is the best? II

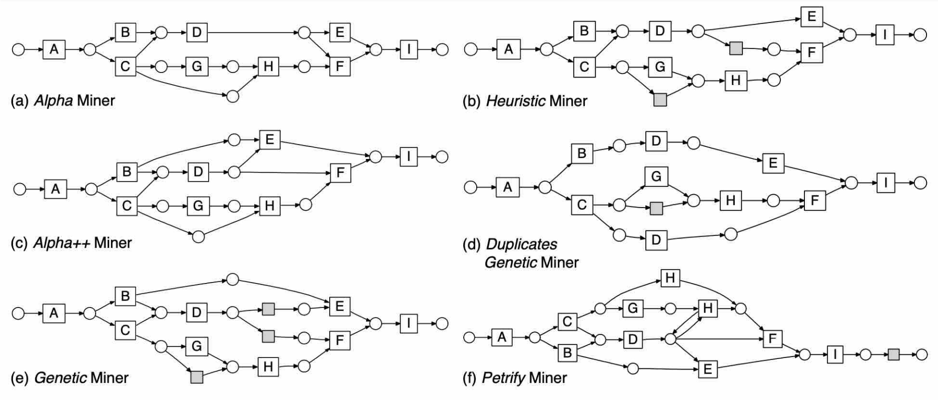


Fig. 1. Process models that were discovered by different process discovery algorithms based on the same log

Which Process Discovery algorithm is the best? III

- Use metrics to evaluate the four quality criteria
- For unstructured processes we have:

	Lookup			*Exploratory*		
	F	P	G	F	P	G
Alpha ++	0.00	0.00	0.00	0.00	0.00	0.00
Heuristic Miner	0.00	0.00	0.00	0.00	0.00	0.00
Inductive Miner	**0.9886**	0.2391	0.9994	**0.9315**	0.1437	0.9992
Genetic Miner	0.9992	0.1800	0.9938	0.6232	0.8053	0.9963
Language Based Regions	0.6163	0.3825	0.9793	0.7835	0.1919	0.9622
State Based Regions	0.8995	0.4233	0.9957	0.9560	0.2942	0.9918
Fuzzy Miner	–	–	–	–	–	–

Process Discovery algorithms : Qualitative comparison I

	Techniques	Visualisation
Alpha ++	Statistics	Petri nets
Heuristic Miner	Statistics/heuristics	Dependency graphs → Causal nets → Petri nets
Inductive Miner	Divide & conquer	Directly follows graphs → Process trees → Petri nets
Genetic Miner	Genetic algorithm	Causal nets → Petri nets
Language Based Regions	Linear programming	Languages process → Petri nets
State Based Regions	Traces abstraction	Transition systems → Petri nets
Fuzzy Miner	Statistics/heuristics	Fuzzy models

Process Discovery algorithms : Qualitative comparison II

	Logs			Model	
	Noise	Incompleteness	Real log	Soundness	Choice and parallelism
Alpha ++	X	X	X	X	X
Heuristic Miner	✓	X	✓	X	X
Inductive Miner	✓	✓	✓	✓	✓
Genetic Miner	✓	✓	✓	✓	X
Language Based Regions	X	X	X	✓	X
State Based Regions	X	X	X	✓	X
Fuzzy Miner	✓	✓	✓	X	X

D'ici, on voit + loin !

La Rochelle Université

univ-larochelle.fr

Analysis of Business Processes

R. Champagnat, M. Trabelsi, A. Hamdi et al.

2023-2024

La Rochelle Université

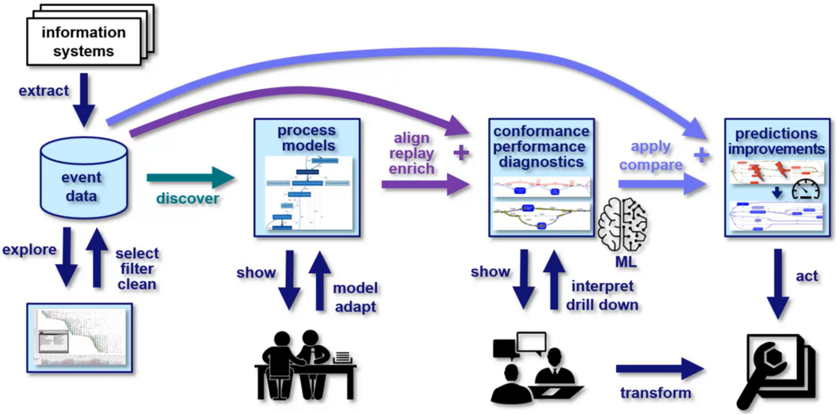
Week 36	Introduction
Week 37	Process discovery (α-Algorithm)
Week 38	Metrics and quality of discovered models
Week 39	Raw traces/ modelled traces (case study)
Week 40	Advanced process mining algorithms
Week 41	Advanced process mining algorithms
Week 42	Conformance checking
Week 46	Decision mining in processes
Week 47	Trace clustering
Week 48	Trace profile
Week 49	Case study
Week 50	Case study defense

Outline

Outline

1. Introduction
2. Model-Based Process Analysis
3. Event Data Analysis
4. Conformance Checking
5. Business Process Analysis
6. Specific cases

Business Process Analysis I



Business Process Analysis II

The classical approach for business process analysis is make up of 5 steps:

1. Obtain an event log
2. Create or discover a process model
3. Connect events, this step is essential for projecting information onto models and to add perspectives
4. Extend the model (add time perspective, connect activities to group of resources, etc.)
5. Return integrated model.

Business Process Analysis III

The technique, which mostly relies on conformance checking, enables:

Check the conformance to specification (audit)

- Audits are carried out to determine the accuracy and dependability of data concerning businesses and the processes that are connected to them.
- Check constraints that management, governments, and other stakeholders have established.

Determine the trace equivalence.

- Two transition systems are equivalent if their sets of execution sequences are identical.
- It uses bisimulation equivalence, or bisimilarity for short. It is a more refined notion taking into account the moment of choice.

Outline

Model-Based Process Analysis I

- **Verification**: it concerns
  - Soundness, completeness, deadlocks...
  - Temporal logic
- **Performance**: three typical dimensions of performance are identified. For each of them different Key Performance Indicators (KPIs) can be defined:
  - Time
    - Lead time (also referred to as flow time) is the total time from the creation of the case to the completion of the case
    - Service time is the time actually worked on a case
    - Waiting time is the time a case is waiting for a resource to become available
    - Synchronization time is the time an activity is not yet fully enabled and waiting for an external trigger or another parallel branch
  - Cost
  - Quality (focuses on the “product” or “service” delivered to the customer)

Outline

Event Log Analysis I

Up to now we focus on case, activity and timestamp. Let us focus on other **event** attribues (resource, costs, ...).

- Important to attach the context to the event (or use trace)
- Use **Dotted Chart** to get an overview of all events
- Use **Visual Inductive Miner** to get an overview of business process execution

Event Log Analysis II

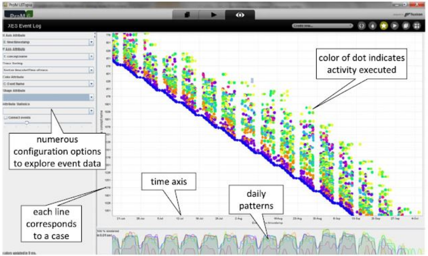


Fig. 11.5 ProM's dotted chart can be used to explore the event data from different angles

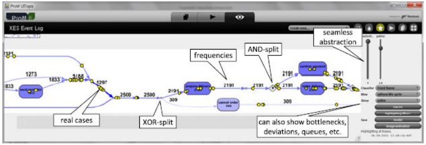


Fig. 11.6 Visual inductive miner replaying the event log on the discovered process model

PM4PY Statistics

PM4PY provides a set of statistics:

- **Throughput Time** (list of all the durations of the cases)
- **Case Arrival/Dispersion Ratio**
- **Cycle Time and Waiting Times**
- **Sojourn Time**
- **Concurrent Activities**
- **Events Distributions**
- **Detection of Batches** (We say that an activity is executed in batches by a given resource when the resource executes several times the same activity in a short period of time.)
- **Rework** (activities, cases): identify the activities which have been repeated during the same process execution.

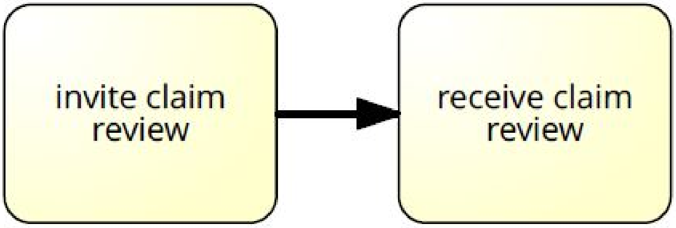
Outline

Introduction to Conformance Checking

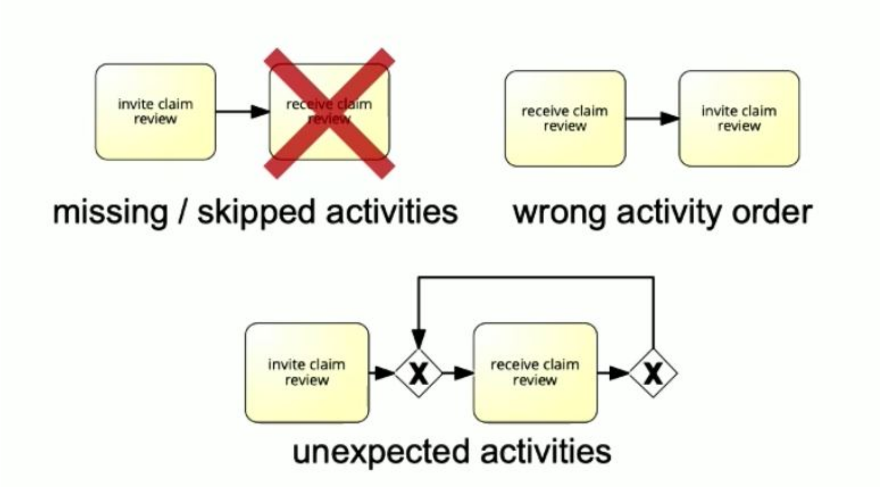
- Compare **event log** to the discovered process models (sometimes to the **blueprint** process model).
- Related to **Fitness** measures (the proportion of behavior in the event log possible according to the model).
- Investigate where the actual process execution **deviates** from the event logs or the plan.
- Most common use case in practice: **identify violation patterns**.

Conformance Checking insights

There are number of violation patterns a conformance analysis can reveal



Conformance Checking insights



Conformance Checking why ?

There are many scenarios, in which a conformance assessment is important:

- Detecting **problems** and **quality improvement potential** in the process (see Quality metrics course).
- Obtaining **feedback** on how well the process is aligned with **expectations** / the intended process.
- Complying with **laws** and **regulations**

Matrice Footprint I

Table 8.4 Differences between the footprints of L_{full} and N_2. The event log and the model "disagree" on 12 of the 64 cells of the footprint matrix

	a	b	c	d	e	f	g	h
a				$\rightarrow$: #				
b				$\|$:$\rightarrow$	$\rightarrow$: #			
c				$\|$:$\rightarrow$	$\rightarrow$: #			
d	$\leftarrow$: #	$\|$:$\leftarrow$	$\|$:$\leftarrow$			$\leftarrow$: #		
e		$\leftarrow$: #	$\leftarrow$: #					
f				$\rightarrow$: #				
g								
h								

Matrice Footprint II

- Conformance analysis based on footprints is only meaningful if the log is complete with respect to the “directly follows” relation
- Does not take into account the number of cases
- Can also be used for log-to-log comparison (detect changes/deviations in the process) and model-to-model comparison (model similarity)

Token-Based Replay

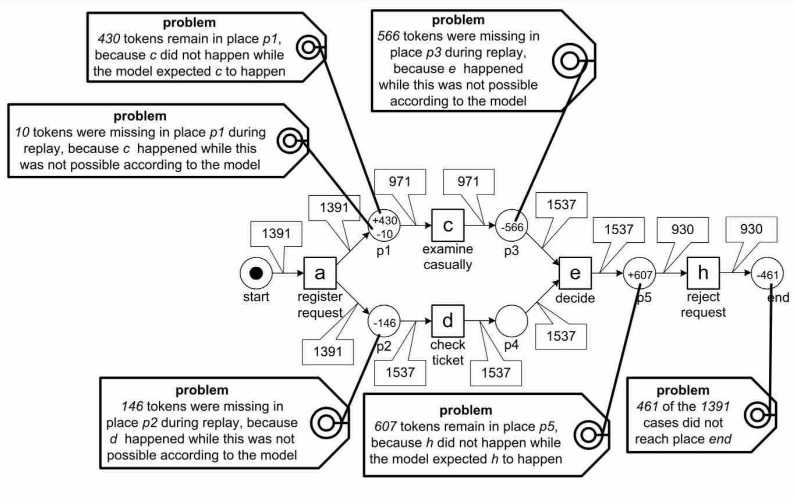


Fig. 8.7 Diagnostic information showing the deviations ($fitness(L_{full}, N_3) = 0.8797$)

Correspondence between process execution and process model

Reminder

In J.E. Cook and A.L. Wolf. Software Process Validation: Quantitatively Measuring the Correspondence of a Process to a Model. ACM Transactions on Software Engineering and Methodology (TOSEM), 8:147–176, April 1999, Cook and Wolf aims to measures the level of correspondence between a process execution and a process model.

154 • J. E. Cook and A. L. Wolf

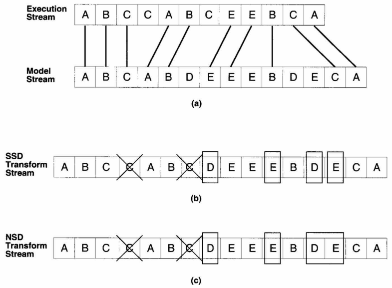
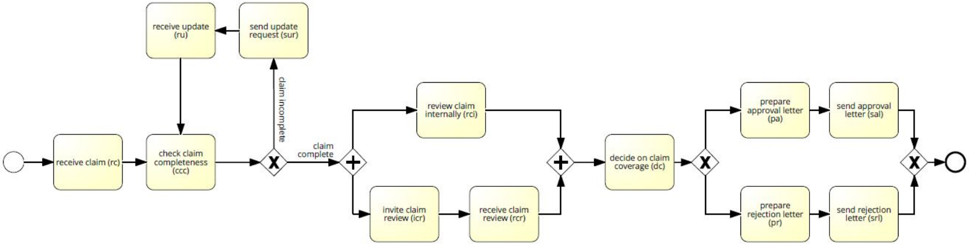


Fig. 3. Example execution and model event streams (a), with execution stream transformed for the SSD metric (b) and the NSD metric (c) calculations.

Alignment

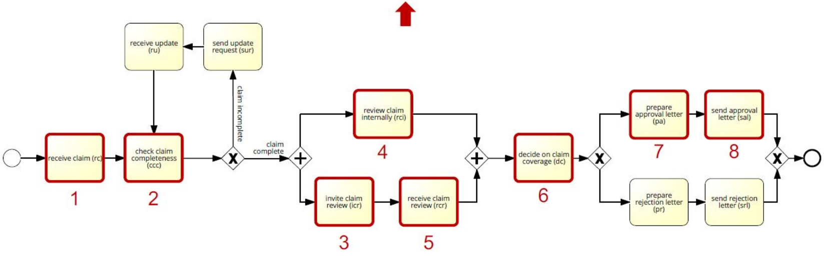
- **How many traces are allowed according to this model?**
- $< rc, ccc, icr, rci, rcr, dc, pa, sal >$
- $< rc, ccc, rci, icr, rcr, dc, pa, sal >$
- $< rc, ccc, icr, rci, rcr, dc, pr, srl >$
- $< rc, ccc, rci, icr, rcr, dc, pr, srl >$ ...



Alignment : Comparing a trace and a model

- **We can detect violation patterns by comparing a considered trace from the event log with the closest trace from the model**
- $< rc, ccc, icr, rci, rcr, dc, pa, sal >$

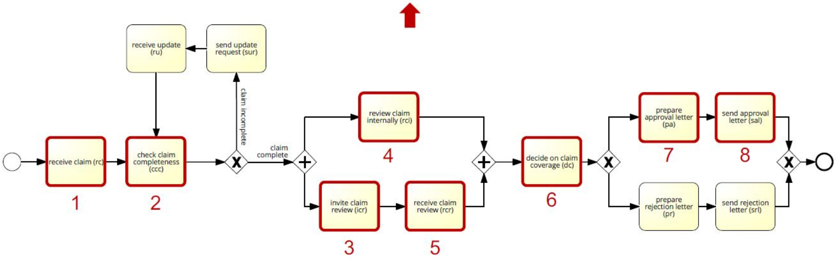
event log trace	rc	ccc	icr	rci	rcr	dc	pa	sal
model trace	rc	ccc	icr	rci	rcr	dc	pa	sal



Alignment : Comparing a trace and a model

- $< rc, ccc, icr, rci, dc, rcr, pa >$

event log trace	rc	ccc	icr	rci	X	dc	rcr	pa	X
model trace	rc	ccc	icr	rci	rcr	dc	X	pa	sal

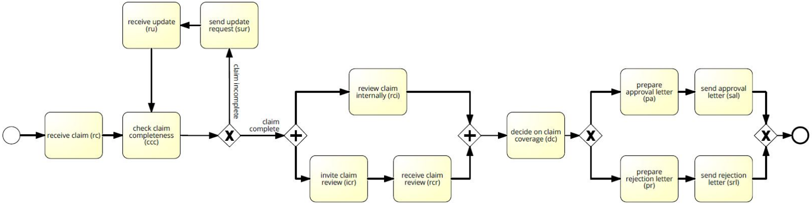


Alignment : From comparison to violation patterns

event log trace	rc	ccc	icr	rci	X	dc	rcr	pa	X
model trace	rc	ccc	icr	rci	rcr	dc	X	pa	sal

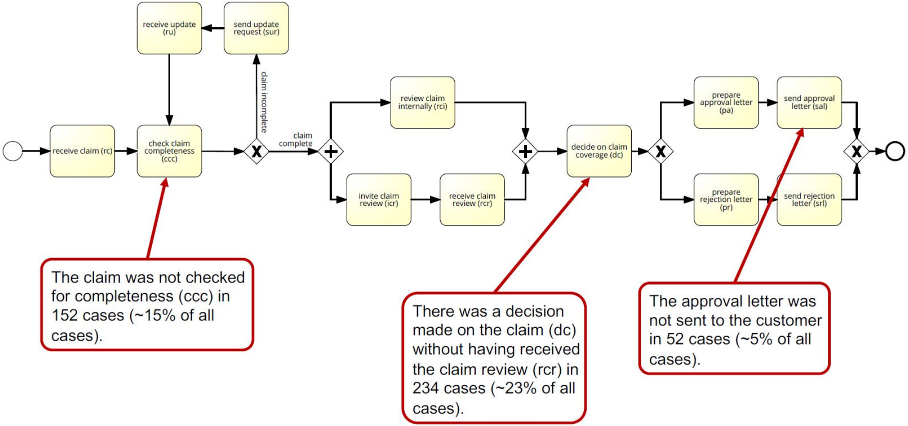
The decision on the claim (dc) was taken before the claim was received (rcr). The activities have been executed in the **wrong order**.

The approval letter was not sent as expected. The activity was **skipped**.



Alignment : From comparison to violation patterns

Imagine we check a log with 1000 traces ...



Trace Alignment : Formally I

- Let Σ denote the set of activities. $|\Sigma|$ is the number of activities.
- Σ^+ is the set of all non-empty finite sequences of activities from Σ. $T \in \Sigma^+$ is a trace over Σ. $|T|$ denotes the length of trace T
- The set of all n-length sequences over the alphabet Σ is denoted by Σ^n. A trace of length n is denoted as T^n i.e., $T^n \in \Sigma^n$, and $|T^n| = n$.
- The ordered sequence of activities in T^n is denoted as $T(1)T(2)T(3)...T(n)$ where $T(k)$ represents the k^{th} activity in the trace
- T^{n-1} denotes the $n-1$ length prefix of T^n. In other words $T^n = T^{n-1}T(n)$
- An event log, $\mathcal{L}$, corresponds to a multi-set (or bag) of traces from Σ^+.

Trace Alignment : Formally II

Bose et al. **Trace Alignment in Process Mining: Opportunities for Process Diagnostics**.

Trace alignment

Trace alignment over a set of traces $\mathbb{T} = \{T_1, T_2, ..., T_n\}$ is defined as a mapping of the set of traces in $\mathbb{T}$ to another set of traces $\overline{\mathbb{T}} = \{\overline{T_1}, \overline{T_2}, ..., \overline{T_n}\}$ where each $\overline{T_i} \in (\Sigma \cup \{-\})^+$ for $1 \leq i \leq n$ and

- $|\overline{T_1}| = |\overline{T_2}| = ... = |\overline{T_n}| = m$,
- $\overline{T_i}$ by removing all "$-$" gap symbols is equal to T_i,
- $\nexists k, 1 \leq k \leq m$ such that $\forall_{1 \leq i \leq n}, \overline{T_i}(k) = -$

For instance, with $\Sigma = \{a, b, c, d, e\}$ we can have $\overline{T_1} = \langle a, -, d, b \rangle$

Paire-wize aligment I

Aligning a pair of traces is referred to as pair-wise trace alignment.
Let us consider the example of aligning the two traces $T_1 = \langle a, b, c, a, c \rangle$ and $T_2 = \langle a, c, a, c, a, d \rangle$. We have three possible alignments:

$$\begin{array}{c|ccccccc} T_1 & a & b & c & a & c & - & - \\ \hline T_2 & a & - & c & a & c & a & d \end{array} \qquad \begin{array}{c|cccccc} T_1 & a & b & c & a & c & - \\ \hline T_2 & a & c & a & c & a & d \end{array}$$

$$\begin{array}{c|ccccccccccc} T_1 & a & b & c & a & c & - & - & - & - & - & - \\ \hline T_2 & - & - & - & - & - & a & c & a & c & a & d \end{array}$$

Paire-wize aligment II

Alignment between a pair of traces, T_1 and T_2 can be considered as a transformation of the trace T_1 to T_2 or vice versa through a set of editing operations applied to one of the traces iteratively. Assuming that $\overline{T_1}$ is written over $\overline{T_2}$ in the alignment the following edit operations are defined for any column j in the alignment:

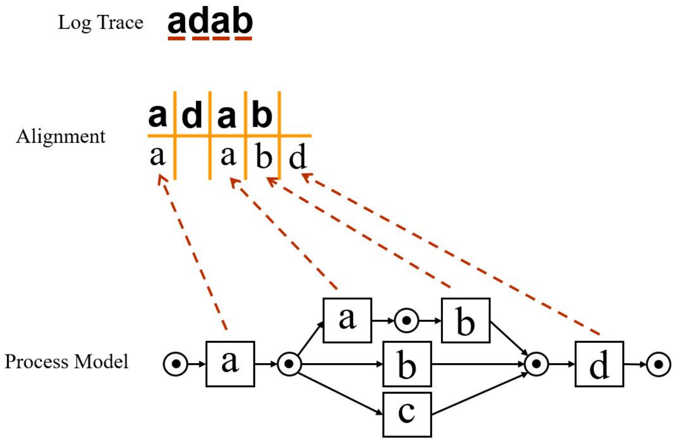
- the activity pair $(a, b), a, b \in \Sigma$, denotes a substitution of activity a in T_1 with activity b in T_2,
- the activity pair $(a, -)$ denotes the deletion of activity a in T_1, and
- the activity pair $(-, b)$ denotes the insertion of activity b in T_1.

Alignment-based Conformance Checking I

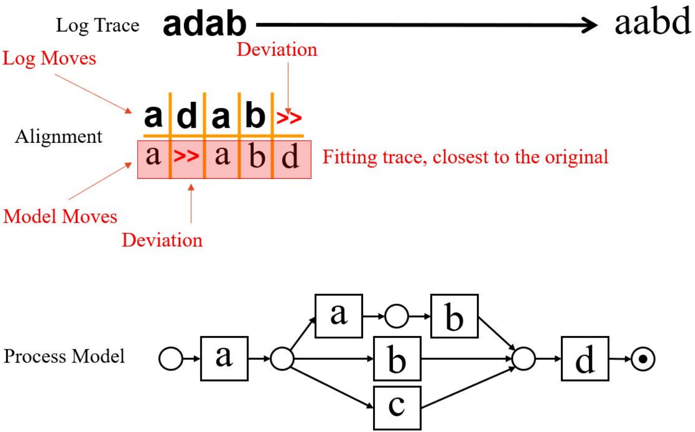
A. Adriansyah. **Aligning Observed and Modeled Behavior**. Phd thesis, Eindhoven University of Technology, April 2014.

- Find the **closest** model trace in the model behavior for a given log trace
- Define an alignment between logs and a process model as a pairwise comparison between executed activities in the logs and the activities allowed by the model.
- Given a **trace** and a **Petri net**, if the trace **perfectly fits** the net each activity in the trace can be mimicked by firing a transition in the net. Furthermore, at the end of the trace the final state should have been reached.

Alignment-based Conformance Checking II



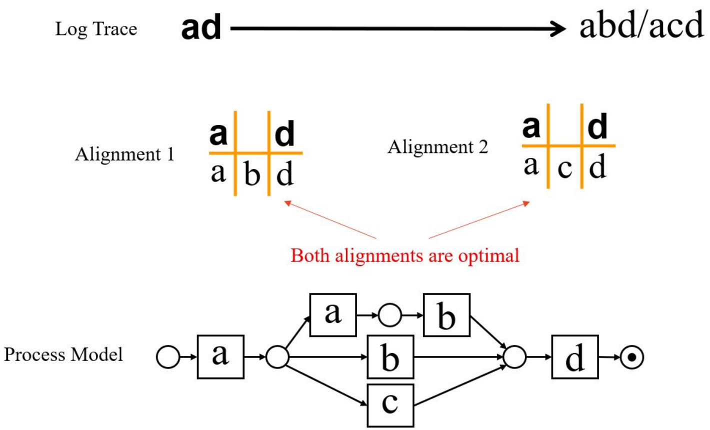
Alignment-based Conformance Checking III



Alignment-based Conformance Checking IV

- **Synchronous Move** : a step in which the event in the trace and the task in the execution sequence of the model correspond to each other.
- **Model Move** : when a task and thus an activity should have been executed according to the model, but there is no related event in the trace.
- **Log Move** : when an event in the trace indicates that an activity has been executed, even though it should not have been executed according to the model.

Alignment-based Conformance Checking V



Alignment-based Conformance Checking VI

A move is a pair $(x, (y, t))$ where the first element refers to the log and the second element refers to the model.

For example,

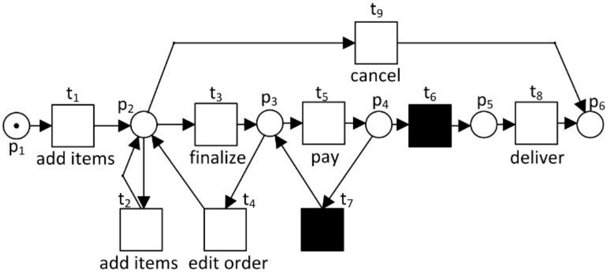


Figure 3.4: An online transaction for an electronic bookstore in Petri net.

Alignment-based Conformance Checking VII

$$\gamma_1 = \begin{array}{|c|c|c|}\hline \textit{add items} & \textit{add items} & \textit{cancel} \\ \hline \textit{add items} & \textit{add items} & \textit{cancel} \\ t_1 & t_2 & t_9 \end{array}$$

Figure 3.5: An alignment between $\sigma_1 = \langle \textit{add items}, \textit{add items}, \textit{cancel} \rangle$ and the model in Figure 3.4.

$$\gamma_2 = \begin{array}{|c|c|c|c|c|c|c|c|}\hline \textit{add items} & \textit{cancel} & \textit{finalize} & \gg & \textit{finalize} & \textit{pay} & \gg & \textit{deliver} \\ \hline \textit{add items} & & \textit{finalize} & \textit{edit order} & \textit{finalize} & \textit{pay} & & \textit{deliver} \\ t_1 & \gg & t_3 & t_4 & t_3 & t_5 & t_6 & t_8 \end{array}$$

Figure 3.6: An alignment between $\sigma_3 = \langle \textit{add items}, \textit{cancel}, \textit{finalize}, \textit{finalize}, \textit{pay}, \textit{deliver} \rangle$ and the model in Figure 3.4.

- $(\textit{additems}, (\textit{additems}, t_1))$ means that both log and model make an "*additems* move" and the move in the model is caused by the occurrence of transition t_1.
- $(\gg, (\textit{editorder}, t_4))$ means that the occurrence of transition t_3 with label *editorder* is not mimicked by a corresponding move of the log.
- $(\textit{cancel}, \gg)$ means that the log makes an "*cancel* move" not followed by the model.

Alignment-based Conformance Checking VIII

$(x, (y, t))$ is a legal move if one of the following four cases holds:

- $x = y$ and y is the visible label of transition t (synchronous move),
- $x = \gg$ and y is the visible label of transition t (visible model move),
- $x = \gg$, $y = \tau$ and transition t is silent (invisible model move), or
- $x \neq \gg$ and $(y, t) = \gg$ (log move).

Other moves such as $(\gg, \gg)$ and $(x, (y, t))$ with $x \neq y$ are illegal moves.

Alignment-based Conformance Checking IX

Alignment

Let $A \subseteq \mathcal{A}$ be a set of activities. Let $\sigma \in A^*$ be a trace over A and let $N = (P, T, F, \alpha, m_i, m_f)$ be a Petri net over A. An alignment $\gamma \in (A^{\gg} \times T^{\gg})^*$ between σ and N is a legal movement sequence such that:

- $\pi_1(\gamma)_{\downarrow A} = \sigma$, i.e. its sequence of movements in the trace (ignoring $\gg$) yields the trace, and
- $m_i \xrightarrow{\pi_2(\gamma)_{\downarrow T}} m_f$, i.e. its sequence of movements in the model (ignoring $\gg$) yields a complete firing sequence of N.

$\Gamma_{\gamma,N}$ is the set of all alignments between a trace σ and a Petri net N.

Note that alignments require termination of both trace and process model.

Cost of Alignment I

We are interested in alignments with the least total likelihood cost according to the assigned likelihood cost function. Such an alignment is called an optimal alignment.

Standard likelihood cost function

Let $A \subseteq \mathcal{A}$ be a set of activities. Let $N = (P, T, F, \alpha, m_i, m_f)$ be a Petri net over A. The standard likelihood cost function $lc : A^{\gg} \times T^{\gg} \to \mathbb{R}$ is the function that maps all movements to real values, such that for all $(x, y) \in A^{\gg} \times T^{\gg}$:

- $lc((x, y)) = 0$ if either $x \in A$, $y \in T$, and $x = \alpha(y)$, or $x = \gg$, $y \in T$, and $\alpha(y) = \tau$,
- $lc((x, y)) = +\infty$ if either $x \in A$, $y \in T$, and $x \neq \alpha(y)$, or $x = y = \gg$, and
- $lc((x, y)) = 1$ otherwise.

Cost of Alignment II

- Assign **zero** cost to all **synchronous moves** of activities and transitions with the same label, as well as to all **moves on model of invisible transitions**.
- Assign cost 1 to all **moves on log/moves** on model of normal (not invisible) transitions.
- $+\infty$ to all **synchronous moves** whose transitions have **different labels** than their activities.

Fitness based optimal alignment computation I

- The alignment technique aims to find the **optimal alignment** with the **lower cost**.
- It affects a standard positive cost for any type of move (*i.e.* $\gg$ symbols). in case of multiple alignments,
- the **Fitness** metric is calculated on each alignment and the optimal one with the best Fitness will be considered:

$$TraceFitness(t, M) = 1 - \frac{\delta(\lambda^M_{opt}(t))}{\delta(\lambda^M_{worst}(t))} \quad (1)$$

where, δ is the cost function, $\lambda^M_{worst}(t)$ is the worst case where there are no synchronous moves between the trace t and the process model M and $\lambda^M_{opt}(t)$ are each cost obtained on each optimal alignment.

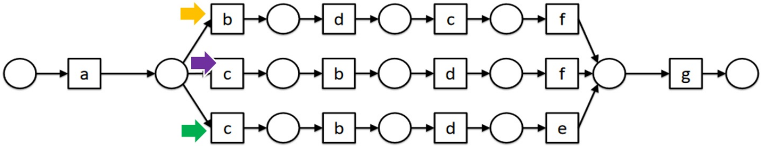
Fitness based optimal alignment computation II

Trace
<a,b,c,d,e,g>

Move log cost =6

Model

Move model cost =6



Trace	a	b	c	d	>>	e	>>	g
Modèle	a	b	>>	d	c	>>	f	g

Cost = 4

Fitness = 1- (4/(6+6)) = 0,67

Trace	a	>>	b	c	d	e	>>	g
Modèle	a	c	b	>>	d	>>	f	g

Cost = 4

Fitness = 0,67

Trace	a	>>	b	c	d	e	g
Modèle	a	c	b	>>	d	e	g

Cost = 2

Fitness = 1- (2/(6+6)) = 0,83

Oracle function I

- An **oracle function** maps each **trace** in the log to **a set of alignments** relating traces to paths in the model.
- For any observed behavior a suitably chosen path through the model is returned.
- An oracle function may use a likelihood cost function to assign probabilities of alignments (may also need to look at the value of these attributes).
- The higher the probability of an alignment of a trace, the more likely the alignment is the “best” representation of the trace.
- An oracle function gives the probabilities of **all possible alignments** between a **given trace** and **Petri net**.

Alignment Quality I

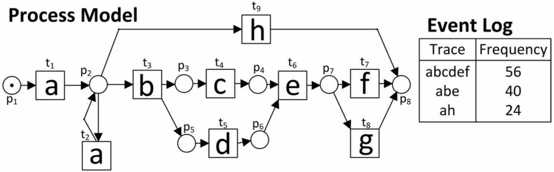


Figure 6.2: Left: The model of an online transaction in an electronic bookstore, shown previously in Figure 4.10 with relabeled activities. **Right:** an event log of the model.

Table 6.1: All optimal alignments between all traces and net N in Figure 6.2 using the standard likelihood cost function.

σ	$\Gamma^o_{\sigma,N}$	Label
$\langle a,b,c,d,e,f\rangle$	$\langle (t_1,a),(t_3,b),(t_4,c),(t_5,d),(t_6,e),(t_7,f)\rangle$	γ_1
$\langle a,b,e\rangle$	$\langle (a,t_1),(b,t_3),(\gg,t_4),(\gg,t_5),(e,t_6),(\gg,t_7)\rangle$	γ_2
	$\langle (a,t_1),(b,t_3),(\gg,t_5),(\gg,t_4),(e,t_6),(\gg,t_7)\rangle$	γ_3
	$\langle (a,t_1),(b,t_3),(\gg,t_4),(\gg,t_5),(e,t_6),(\gg,t_8)\rangle$	γ_4
	$\langle (a,t_1),(b,t_3),(\gg,t_5),(\gg,t_4),(e,t_6),(\gg,t_8)\rangle$	γ_5
	$\langle (a,t_1),(\gg,t_9),(b,\gg),(e,\gg)\rangle$	γ_6
	$\langle (a,t_1),(b,\gg),(e,\gg),(\gg,t_9)\rangle$	γ_7
	$\langle (a,t_1),(b,\gg),(\gg,t_9),(e,\gg)\rangle$	γ_8
$\langle a,h\rangle$	$\langle (a,t_1),(h,t_9)\rangle$	γ_9

$$\gamma_x = \begin{array}{|c|c|c|c|c|c|} a & b & c & d & e & \gg \\ \hline a & b & c & d & e & f \\ t_1 & t_3 & t_4 & t_5 & t_6 & t_7 \end{array} \qquad \gamma_y = \begin{array}{|c|c|} a & \gg \\ \hline a & h \\ t_1 & t_9 \end{array}$$

Figure 6.3: Left: An optimal alignment between $\sigma_x = \langle a,b,c,d,e\rangle$ and the model of Figure 6.2, **Right:** An optimal alignment between $\sigma_y = \langle a\rangle$ and the same model.

Both optimal alignments show exactly one deviation. γ_y is much shorter than γ_x. Intuitively, the quality of γ_x should be better than γ_y.

Alignment Quality II

Therefore, when comparing two alignments computed from two different traces and the same Petri net, we also take into account the length of the traces.

Alignment Quality

Let $A \subseteq \mathcal{A}$ be a set of activities. Let $\sigma \in A^*$ be a trace over A and let $N = (P, T, F, \alpha, m_i, m_f)$ be a sound Petri net over A. Let $lc : (A^{\gg} \times T^{\gg}) \rightarrow \mathbb{R}$ be a likelihood cost function for movements. The quality of alignment $\gamma \in \Gamma_{\sigma,N}$ with respect to likelihood function lc is:

$$aql(\gamma, N, lc) = 1 - \frac{\Sigma_{(x,y)\in\gamma} lc((x, y))}{lim(\pi_1(\gamma)_{\downarrow A}, N, lc)} \quad (2)$$

where lim is the likelihood cost limit between σ and N with respect to lc.

Summary

The goal of conformance checking is to underline similarities and differences between a modelled behavior (process models) and an observed behaviour (event logs).

- Attention to the size of event log when applying conformance checking
- Other Applications of Conformance Checking:
  - Repairing Models
  - Evaluating Process Discovery Algorithms
  - Connecting Event Log and Process Model

Alignment in Pm4py

Alignments in pm4py

Like token-based-replay, computing alignments in `pm4py` is rather straightforward:

```
In [6]:
pn, im, fm = pm4py.discover_petri_net_inductive(df)
pm4py.conformance_diagnostics_alignments(df_problems, pn, im, fm)
```

```
aligning log, completed variants ::   0%|          | 0/6 [00:00<?, ?it/s]

Out[6]: [{'alignment': [('>>', 'register request'),
    ('>>', None),
    ('examine thoroughly', 'examine thoroughly'),
    ('check ticket', 'check ticket'),
    ('decide', 'decide'),
    ('>>', None),
    ('reject request', 'reject request')],
   'cost': 10002,
   'visited_states': 7,
   'queued_states': 22,
   'traversed_arcs': 22,
   'lp_solved': 1,
   'fitness': 0.8888888888888888,
   'bwc': 90002},
```

Alignment in Pm4py

Like token-based-replay, alignments can also be used to quantify 'fitness':

```
In [7]:  pm4py.fitness_alignments(df_problems, pn, im, fm)

aligning log, completed variants ::   0%|          | 0/6 [00:00<?, ?it/s]

Out[7]: {'percFitTraces': 16.666666666666668,
         'averageFitness': 0.8446623093681916,
         'percentage_of_fitting_traces': 16.666666666666668,
         'average_trace_fitness': 0.8446623093681916,
         'log_fitness': 0.843731055042718}
```

Outline

Key questions

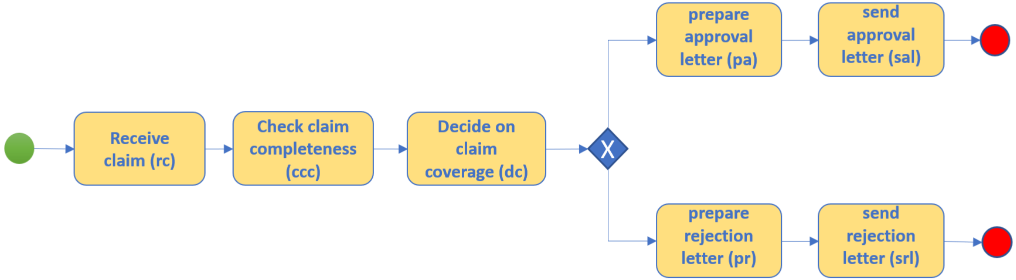
1. What is the most common process behavior that is executed?
2. Where do process instances deviate and what do they have in common?
3. What are the contexts in which an activity is executed?
4. What are the process instances that exactly or approximately capture a desired behavior?
5. Are there particular patterns (e.g., milestones, concurrent activities, etc.) in my process?

Key questions

1. What is the most common process behavior that is executed?
2. Where do process instances deviate and what do they have in common?
3. What are the contexts in which an activity is executed?
4. What are the process instances that exactly or approximately capture a desired behavior?
5. Are there particular patterns (e.g., milestones, concurrent activities, etc.) in my process?
6. Are there any decision points and/or bottlenecks?

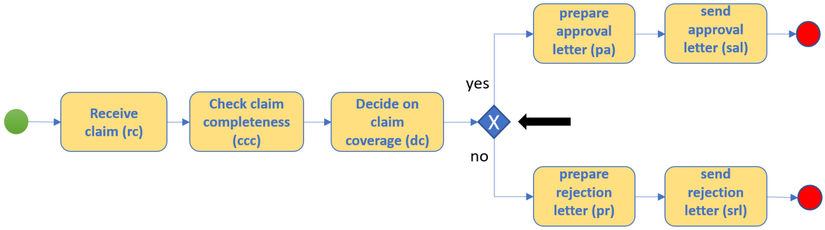
Decision points I

Decision points are OR-split (book flight or book hotel) or XOR-split (accept the claim or reject the claim) in the log.



Decision points I

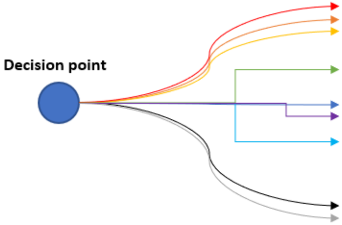
Check	ClaimID	EmpID	Complete	CheckDate	CheckTime
	1239	emp182	yes	06.04.2020	8:23
	1234	emp186	yes	06.04.2020	8:23
	1236	emp184	no	06.04.2020	8:28
	1238	emp120	yes	06.04.2020	8:29
	1235	emp182	yes	06.04.2020	8:29
	1241	emp184	yes	14.04.2020	8:23
	1240	emp120	no	14.04.2020	8:23
	1237	emp182	yes	14.04.2020	8:28
	1244	emp186	yes	14.04.2020	8:29
	1242	emp184	yes	14.04.2020	8:32
	1245	emp120	yes	14.04.2020	8:35
	1243	emp182	yes	14.04.2020	8:40



Decision points I

Decision mining aims to find rules explaining choices in terms of the characteristics of the case

- Classification techniques can be used to find such rules
  - Decision Trees
  - Support Vector Machines
  - Neural Networks
  - ...
- Let us consider a simple approach based on decision trees
  - Decision trees are intuitive and easy to explain
  - Decision trees do not require normalization
  - Decision trees can also deal with missing values



Decision Tree I

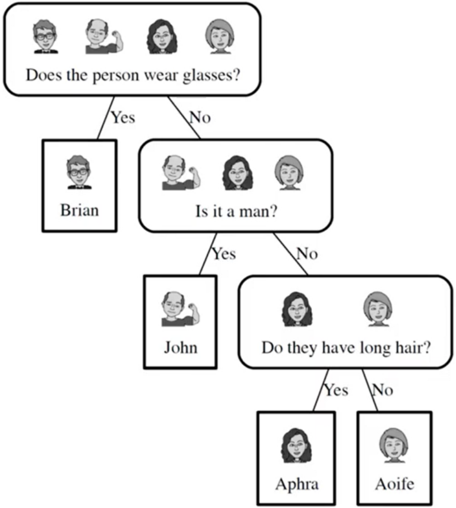
Definition

Decision Tree is a supervised learning technique aiming at the classification of instances based on predictor variables.

- Response variable (dependent variable)
- Predictor variables (independent variables)

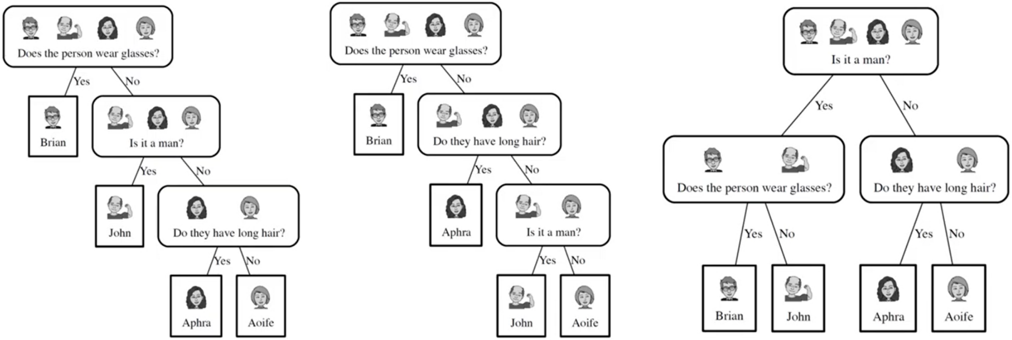
Goal

Partitioning instances in increasingly smaller groups



Decision Tree II

- Many trees are possible:
  1. The tree is small and simple
  2. The leaves are homogeneous in terms of the target feature



Decision Tree III

Decision trees aim to explain the target feature (class) in terms of descriptive features

features

instances

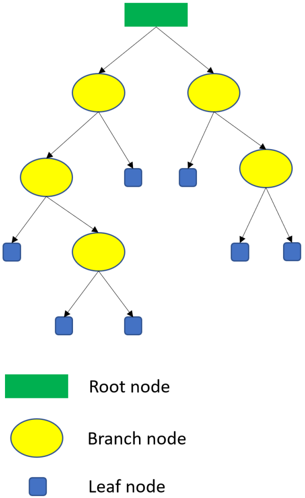
	f_1	f_2	f_3	…	…	f_m	**class**
i_1							Small
i_1							Large
i_3							Medium
…							Small
…							Small
i_n							Medium

Descriptive features

Target feature

Decision Tree IV

- A Decision tree consists of three types of nodes
  1. Root node
  2. Branch node
  3. Leaf node
- Tree generator determines
  1. Which variable to split at a node and what will be the value of the split?
  2. Decision to stop (make a terminal note) or split again has to be made
  3. Assign terminal nodes to a label



Example I

	Feel	Temp.	Humidity	Wind	Play Golf
1	sun	warm	high	false	no
2	sun	warm	high	true	no
3	cloudy	warm	high	false	yes
4	rain	good	high	false	yes
5	rain	cool	normal	false	yes
6	rain	cool	normal	true	no
7	cloudy	cool	normal	true	yes
8	sun	good	high	false	no
9	sun	cool	normal	false	yes
10	rain	good	normal	false	yes
11	sun	good	normal	true	yes
12	cloudy	good	high	true	yes
13	cloudy	warm	normal	false	yes
14	rain	good	high	true	no

Descision Tree : steps

1. calculate the entropy of the whole dataset
2. calculate the entropy of each individual attribute
3. calculate the information gain

Example II

	Feel	Temp.	Humidity	Wind	Play Golf
1	sun	warm	high	false	no
2	sun	warm	high	true	no
3	cloudy	warm	high	false	yes
4	rain	good	high	false	yes
5	rain	cool	normal	false	yes
6	rain	cool	normal	true	no
7	cloudy	cool	normal	true	yes
8	sun	good	high	false	no
9	sun	cool	normal	false	yes
10	rain	good	normal	false	yes
11	sun	good	normal	true	yes
12	cloudy	good	high	true	yes
13	cloudy	warm	normal	false	yes
14	rain	good	high	true	no

Entropy

$$E = -\sum_{i=1}^{k} p_i.log_2(p_i)$$

while $\log_2 x = \frac{\ln x}{\ln 2}$

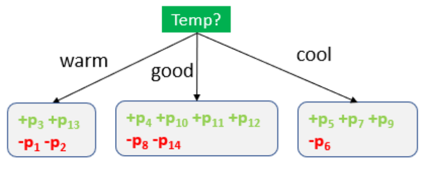
- $E = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.94$

Example III

	Feel	Temp.	Humidity	Wind	Play Golf
1	sun	warm	high	false	no
2	sun	warm	high	true	no
3	cloudy	warm	high	false	yes
4	rain	good	high	false	yes
5	rain	cool	normal	false	yes
6	rain	cool	normal	true	no
7	cloudy	cool	normal	true	yes
8	sun	good	high	false	no
9	sun	cool	normal	false	yes
10	rain	good	normal	false	yes
11	sun	good	normal	true	yes
12	cloudy	good	high	true	yes
13	cloudy	warm	normal	false	yes
14	rain	good	high	true	no

Attribute: Temperature

values(temp.) = sun, cloudy, rain



Example IV

	Feel	Temp.	Humidity	Wind	Play Golf
1	sun	warm	high	false	no
2	sun	warm	high	true	no
3	cloudy	warm	high	false	yes
4	rain	good	high	false	yes
5	rain	cool	normal	false	yes
6	rain	cool	normal	true	no
7	cloudy	cool	normal	true	yes
8	sun	good	high	false	no
9	sun	cool	normal	false	yes
10	rain	good	normal	false	yes
11	sun	good	normal	true	yes
12	cloudy	good	high	true	yes
13	cloudy	warm	normal	false	yes
14	rain	good	high	true	no

Attribute: Temperature

- $E_1 = -\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4} = 1$
- $E_2 = -\frac{4}{6}\log_2\frac{4}{6} - \frac{2}{6}\log_2\frac{2}{6} = 0.918$
- $E_3 = -\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4} = 0.811$

$E = \frac{4}{14}\times 1 + \frac{6}{14}\times 0.918 + \frac{4}{14}\times 0.811 = 0.91$

$GI = 0.94 - 0.91 = \mathbf{0.03}$

Example V

	Feel	Temp.	Humidity	Wind	Play Golf
1	sun	warm	high	false	no
2	sun	warm	high	true	no
3	cloudy	warm	high	false	yes
4	rain	good	high	false	yes
5	rain	cool	normal	false	yes
6	rain	cool	normal	true	no
7	cloudy	cool	normal	true	yes
8	sun	good	high	false	no
9	sun	cool	normal	false	yes
10	rain	good	normal	false	yes
11	sun	good	normal	true	yes
12	cloudy	good	high	true	yes
13	cloudy	warm	normal	false	yes
14	rain	good	high	true	no

Attribute: Feel

values(feel) = sun, cloudy, rain

- $E_1 = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.97$
- $E_2 = -\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4} = 0$
- $E_3 = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.97$

$E = \frac{5}{14}\times 0.971 + \frac{4}{14}\times 0 + \frac{5}{14}\times 0.97 = 0.69$

$GI = 0.94 - 0.69 = \mathbf{0.25}$

Example VI

	Feel	Temp.	Humidity	Wind	Play Golf
1	sun	warm	high	false	no
2	sun	warm	high	true	no
3	cloudy	warm	high	false	yes
4	rain	good	high	false	yes
5	rain	cool	normal	false	yes
6	rain	cool	normal	true	no
7	cloudy	cool	normal	true	yes
8	sun	good	high	false	no
9	sun	cool	normal	false	yes
10	rain	good	normal	false	yes
11	sun	good	normal	true	yes
12	cloudy	good	high	true	yes
13	cloudy	warm	normal	false	yes
14	rain	good	high	true	no

Attribute: Humidity

values(feel) = high, normal

- $E_1 = -\frac{3}{7}\log_2\frac{3}{7} - \frac{4}{7}\log_2\frac{4}{7} = 0.9852$
- $E_2 = -\frac{6}{7}\log_2\frac{6}{7} - \frac{1}{7}\log_2\frac{1}{7} = 0.5916$

$E = \frac{7}{14} \times 0.9852 + \frac{7}{14} \times 0.5916 = 0.79$

$GI = 0.94 - 0.79 = \mathbf{0.15}$

Example VII

	Feel	Temp.	Humidity	Wind	Play Golf
1	sun	warm	high	false	no
2	sun	warm	high	true	no
3	cloudy	warm	high	false	yes
4	rain	good	high	false	yes
5	rain	cool	normal	false	yes
6	rain	cool	normal	true	no
7	cloudy	cool	normal	true	yes
8	sun	good	high	false	no
9	sun	cool	normal	false	yes
10	rain	good	normal	false	yes
11	sun	good	normal	true	yes
12	cloudy	good	high	true	yes
13	cloudy	warm	normal	false	yes
14	rain	good	high	true	no

Attribute: Wind

values(feel) = false, true

- $E_1 = -\frac{6}{8}\log_2\frac{6}{8} - \frac{2}{8}\log_2\frac{2}{8} = 0.8113$
- $E_2 = -\frac{3}{6}\log_2\frac{3}{6} - \frac{3}{6}\log_2\frac{3}{6} = 1$

$E = \frac{8}{14} \times 0.8113 + \frac{6}{14} \times 1 = 0.89$

$GI = 0.94 - 0.89 = \mathbf{0.05}$

Example VIII

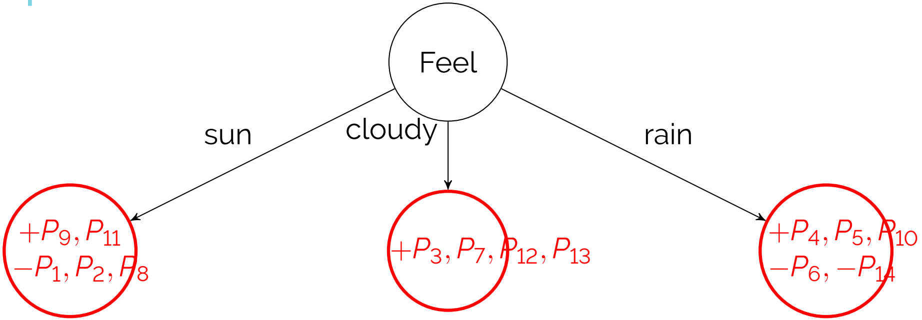
	Feel	Temp.	Humidity	Wind	Play Golf
1	sun	warm	high	false	no
2	sun	warm	high	true	no
3	cloudy	warm	high	false	yes
4	rain	good	high	false	yes
5	rain	cool	normal	false	yes
6	rain	cool	normal	true	no
7	cloudy	cool	normal	true	yes
8	sun	good	high	false	no
9	sun	cool	normal	false	yes
10	rain	good	normal	false	yes
11	sun	good	normal	true	yes
12	cloudy	good	high	true	yes
13	cloudy	warm	normal	false	yes
14	rain	good	high	true	no

Information Gain

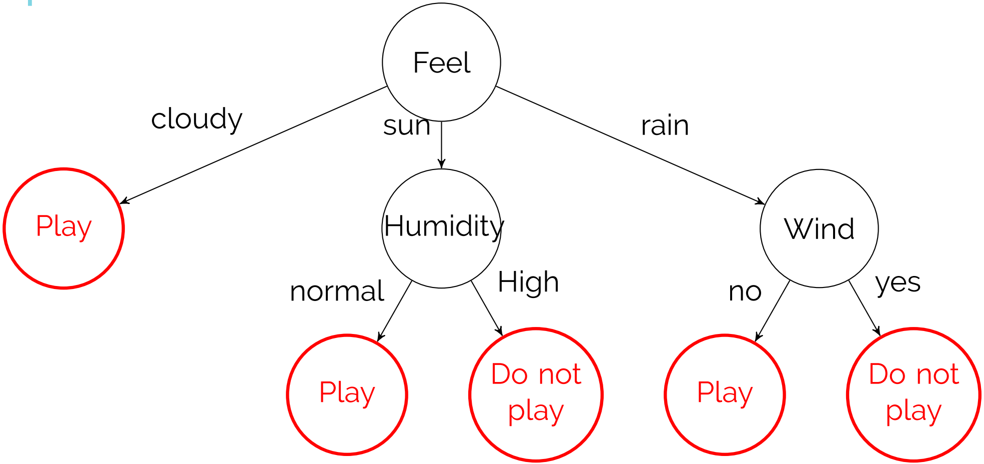
- $GI_{temp} = 0.03$
- $GI_{feel} = 0.25$
- $GI_{humidity} = 0.15$
- $GI_{wind} = 0.05$

Feel is the attribute with the maximum gain $\rightarrow$ Root of the tree

Example IX


Feel
sun
cloudy
rain
$+P_9, P_{11}$
$-P_1, P_2, P_8$
$+P_3, P_7, P_{12}, P_{13}$
$+P_4, P_5, P_{10}$
$-P_6, -P_{14}$

Example X


Feel
cloudy
sun
rain
Play
Humidity
Wind
normal
High
no
yes
Play
Do not play
Play
Do not play

Example XI

Adaptation to Process Mining:

- **Response variable**: the activity executed at decision points (OR-split or XOR-split)
- **Predictor variables**: attributes of events (the context) and/or the previous activities.

Mining Bottlenecks I

Bottlenecks are points of congestion in any process that slow or delay the goal being achieved. **Bottlenecks** are generally one process in a chain of processes, which causes the process to slow down or fail.

Bottlenecks can be:

- **Short-Term**: These are the more obvious problems caused by temporary circumstances. For example, if two employees call in sick and no one else is available to cover their work, a backlog will build until their return.
- **Long-Term**: Long-term bottlenecks are more insidious in nature. They are chronic issues that become accepted as part of the process over time, instead of being identified as an ongoing problem needing a solution.

Mining Bottlenecks II

A **bottleneck** occurs when there is not enough capacity to meet the demand or throughput for a product or service.
How to identify bottlenecks:

- Add timing information to the discovered model
- Identify long wait times or slow processing
- Visual inductive miner supports bottleneck analysis

Mining Bottlenecks III

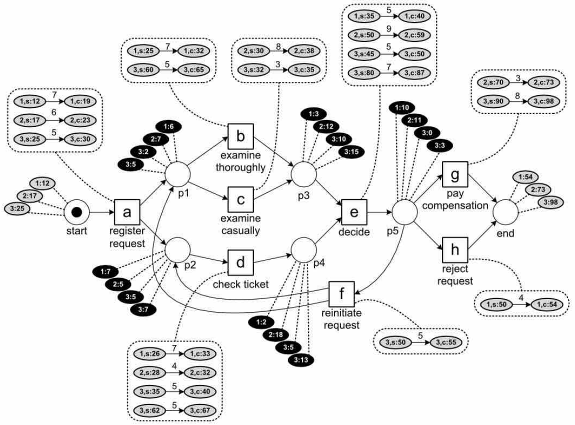


Fig. 9.11 Timed replay of the first three cases in the event log: case 1 starts at time 12 and ends at time 54, case 2 starts at time 17 and ends at time 73, case 3 starts at time 25 and ends at time 98

Mining Bottlenecks IV

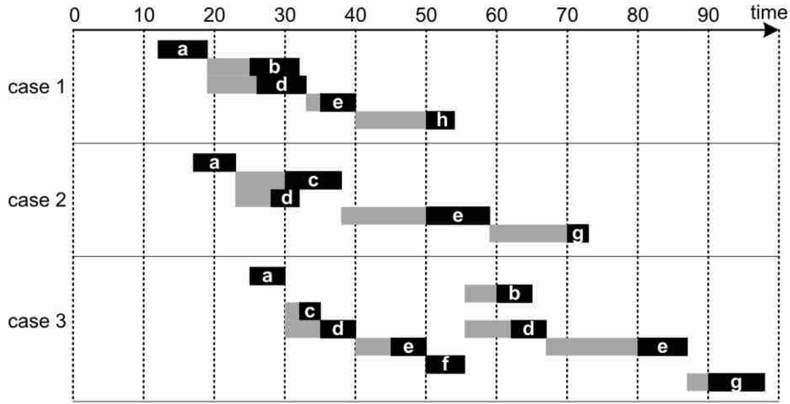


Fig. 9.12 Timeline showing the activity instances of the first three activities

Organizational Mining

Organizational mining focuses on other perspectives:

- **Social**: identify interpersonal relationships in a process (regarding who is performing a process activity and handovers)
- **Organizational structure**
  The behavior of a resource can be characterized by a profile, i.e., a vector indicating how frequently each activity has been executed by the resource. Clustering algorithms can be used to discover similar resources.
- **Resource behaviour**

Taking Decisions Over a Discovered Model I

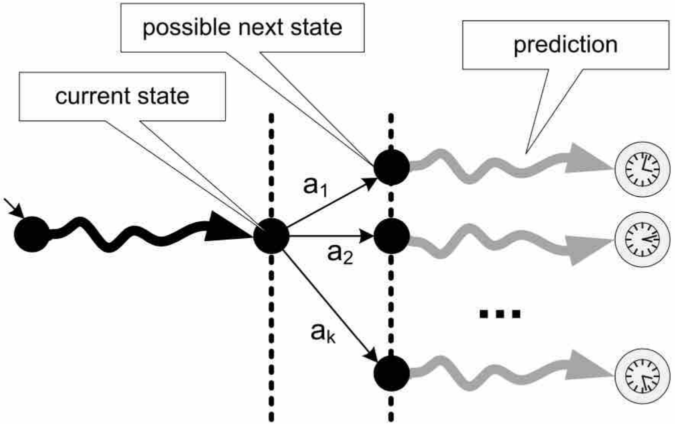
- The model is Digital Twins. It can be used to simulate various parameters to identify the benefits of decisions
- **Explore**: The combination of event data and models can be used to explore business processes at run-time. Running cases can be visualized and compared with similar cases that were handled earlier.
- **Predict**: By combining information about running cases with models (discovered or hand-made), it is possible to make predictions about the future, e.g., the remaining flow time and the probability of success. Various techniques can be used to generate predictions. For example, the supervised learning techniques

Taking Decisions Over a Discovered Model II

- **Recommend**. The information used for predicting the future can also be used to recommend suitable actions (e.g. to minimize costs or time). The goal is to enable functionality similar to the guidance given by car navigation systems.

Taking Decisions Over a Discovered Model III

Fig. 10.12
Recommendations can be based on predictions. For every possible choice, simply predict the performance indicator of interest. Then, recommend the best one(s)



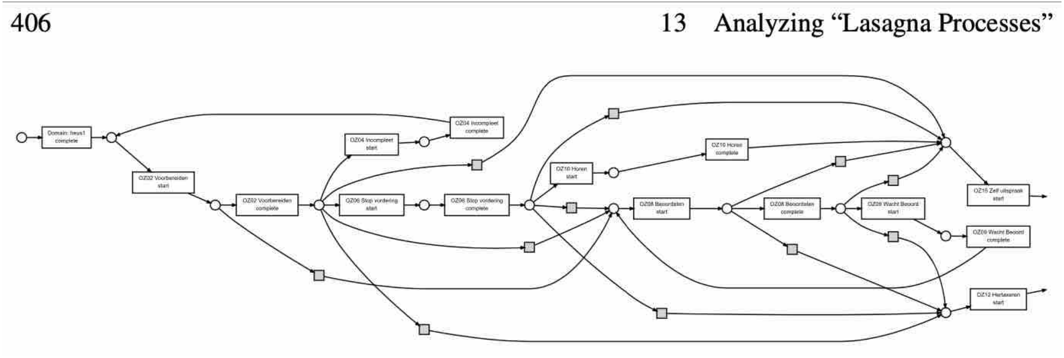
Outline

Italian analysis I

Analyzing “Lasagna Processes"

- Lasagna processes have a clear structure and most cases are handled in a prearranged manner.
- A process is a Lasagna process if with limited efforts it is possible to create an agreed-upon process model that has a fitness of at least 0.8,

Italian analysis II



Italian analysis III

Analyzing "Spaghetti Processes"

- The Spaghetti process comes from unstructured process
- Use clustering

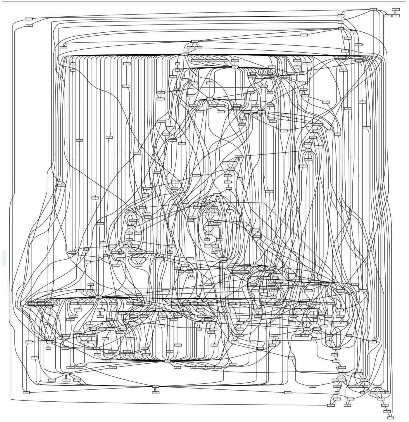


Fig. 14.12 Another Spaghetti process. The model is based on a group of 627 gynecological oncology patients. The event log contains 24331 events referring to 376 different activities


D'ici, on voit + loin !
La Rochelle Université
univ-larochelle.fr

Trace Clustering

M. Trabelsi, N. Joudieh, R. Champagnat et al.

2023-2024

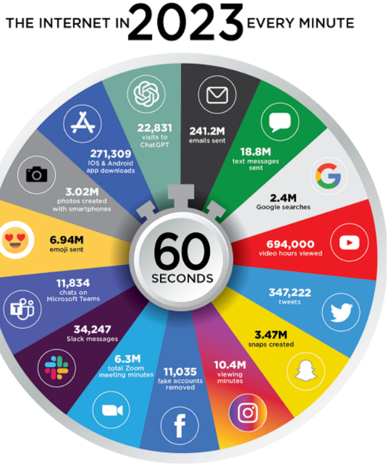
Week 36	Introduction
Week 37	Process discovery (α-Algorithm)
Week 38	Metrics and quality of discovered models
Week 39	Raw traces/ modelled traces (case study)
Week 40	Advanced process mining algorithms
Week 41	Advanced process mining algorithms
Week 42	Conformance checking
Week 46	Decision mining in processes
Week 47	Trace clustering
Week 48	Trace profile
Week 49	Case study
Week 50	Case study defense

Outline

Outline

1. Introduction
2. Clustering Overview
3. Trace clustering
4. Real life example
5. Conclusion

We are drowning in data!



Data Mining

What is Data Mining?

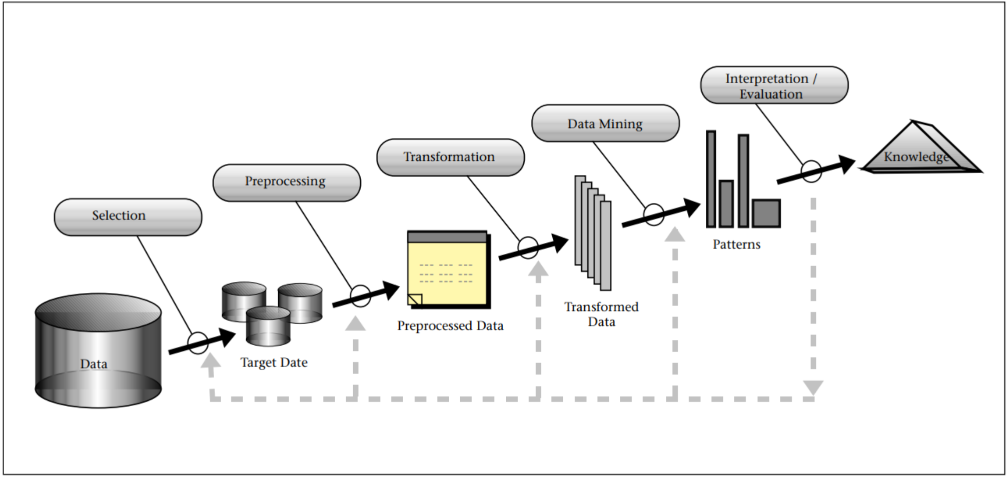
- Huge quantities of data are collected each second
- Data contains interesting patterns
- Patterns are more meaningful and important than data itself
- Data Mining is thus used to:
  - Discover interesting patterns in large quantities of data
  - Support human decision-making provided the discovered patterns

Definition

Data Mining is the exploration and analysis of large quantities of data to discover meaningful patterns. [a]

[a] From *Michael J.A. Berry, Gordon Linoff. Data mining techniques: for marketing, sales, and customer relationship management /2nd ed, 2004*

Data Mining Process



From Fayyad et al. 1996

Data Mining and Machine Learning

Data Mining Tasks and Techniques

- **Descriptive Tasks** *(Unsupervised Learning)*
  - **Goal:** Find patterns in data
  - **Example:**
    - Cluster Analysis or Clustering
    - Association Analysis
- **Predictive Tasks** *(Supervised Learning)*
  - **Goal:** Predict unknown values of a variable, given some observations
  - **Example:**
    - Classification
    - Regression

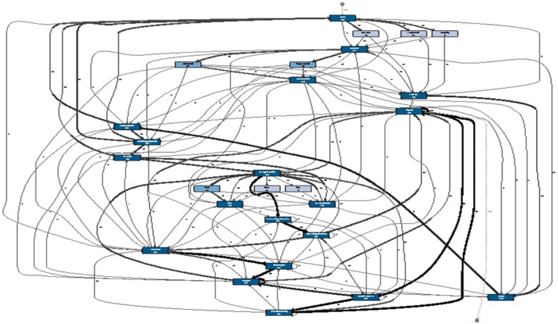
Application Fields

- E-Learning, E-Commerce, Military, Marketing, Health, Fraud Detection, ...

What about huge amounts of traces?

Spaghetti models!!

- Huge amount of data —> Process Mining techniques will discover complex users' behaviors models.
- NEED FOR **CLUSTERING!!**

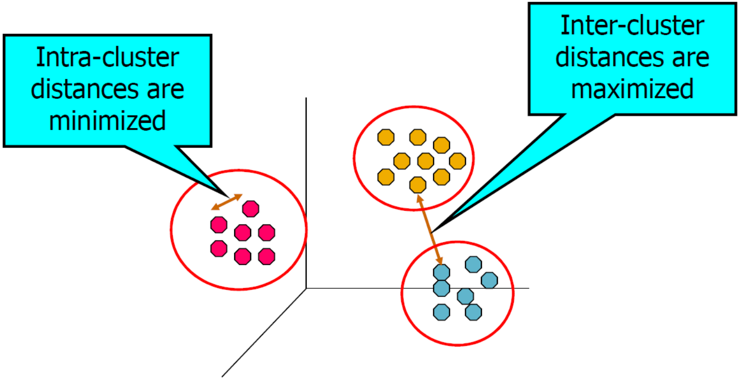


Outline

Clustering

What is Clustering?

Grouping objects such that objects in a group (cluster) are similar to one another **and different from** the objects in other groups (clusters)



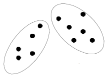
Clustering Algorithms

Clustering Types and Algorithms

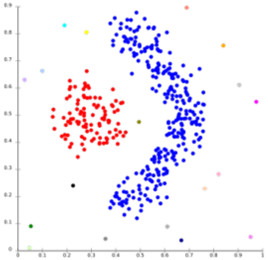
There are different types of clustering:

- **Partitional**
  - Dividing data objects into non-overlapping clusters such that each data object is in exactly one subset
  - Algorithms: K-Means, K-Medoids...
- **Density-Based**
  - Identifying distinctive groups in the data, based on the idea that a cluster in a data space is a contiguous region of high point density, separated from other such clusters by contiguous regions of low point density.
  - Algorithms: DBSCAN, Meanshift, OPTICS, DENCLU,...
- **Hierarchical**
  - A set of nested clusters organized as a hierarchical tree (Dendrogram tree)
  - Algorithms: Agglomerative, Divisive, ...

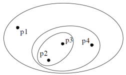
Clustering Types



(a) Partitional Clustering



(b) Density-Based Clustering



(c) Hierarchical Clustering

Clustering Related Aspects

What are the components needed to do clustering?

- Clustering Algorithm:
  - Partitional
  - Density-Based
  - Hierarchical
  - ...
- Proximity Measure (Similarity or Dissimilarity)
  - Euclidean distance
  - Cosine similarity
  - ...
- The Ultimate Goal
  - Minimize **intra-cluster distance**
  - Maximize **inter-clusters distance**
  - Relevance of clustering with analysis aim

Proximity Measures

Proximity Measures

1. Manhattan Distance
2. Euclidean Distance
3. Cosine Measure
4. Jaccard Index
5. Edit Distance - Levenshtein Distance

Distance Measures

Herman Minkowski

Generic Distance Metric for Euclidean and Manhattan.

$$\mathbf{x} = (x_1, x_2, \cdots, x_n) \text{ and } \mathbf{y} = (y_1, y_2, \cdots, y_n)$$

$$d(\mathbf{x}, \mathbf{y}) = \left(|x_1 - y_1|^p + |x_2 - y_2|^p \cdots + |x_n - y_n|^p\right)^{\frac{1}{p}}, \quad p > 0$$

$p = 1$: Manhattan distance

$$d(\mathbf{x}, \mathbf{y}) = |x_1 - y_1| + |x_2 - y_2| \cdots + |x_n - y_n|$$

$p = 2$: Euclidean distance

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{|x_1 - y_1|^2 + |x_2 - y_2|^2 \cdots + |x_n - y_n|^2}$$

Distance Measures

Cosine Measure

- Determines the cosine of the angle between two vectors

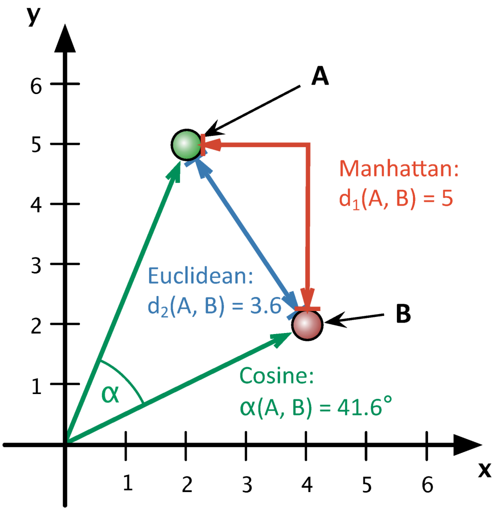
$$\mathbf{x} = (x_1, x_2, \cdots, x_n) \text{ and } \mathbf{y} = (y_1, y_2, \cdots, y_n)$$

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{x_1 y_1 + \cdots + x_n y_n}{\sqrt{x_1^2 + \cdots + x_n^2}\sqrt{y_1^2 + \cdots + y_n^2}}$$

$$d(\mathbf{x}, \mathbf{y}) = 1 - \cos(\mathbf{x}, \mathbf{y})$$

where: $0 \leq d(\mathbf{x}, \mathbf{y}) \leq 2$

Distance Measures



Distance Measures

Jaccard Index

- Measures the similarity of two data sets, as their intersection divided by their union

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

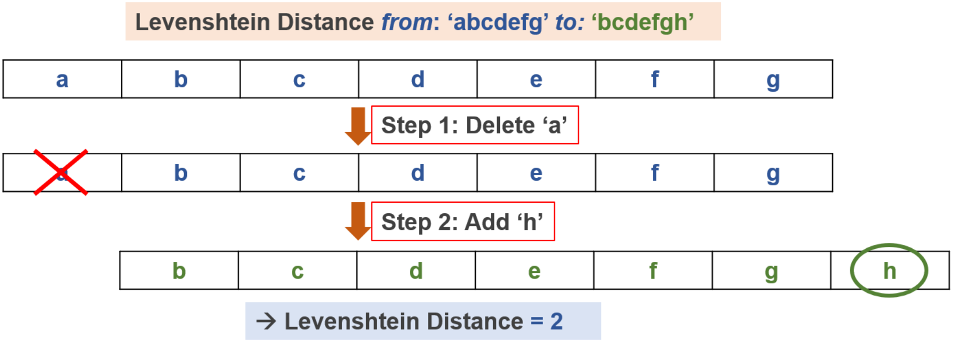
- How to interpret the value of this index?
  - Set a threshold of similarity t
  - if $J(A,B) \geq t$, then sets A and B are said to be similar; else they are not similar

Distance Measures

Edit Distance - Levenshtein distance

- **Edit distance** is a measure that quantifies how dissimilar two sequences are from each other.
  It is measured by counting the number of steps/operations needed to transform one sequence into the other.
- The possible operations are delete, replace, or insert
- **Levenshtein distance**, a type of edit distance, measures the difference between two sequences
- The Levenshtein distance between two sequences is the minimum number of edits needed to change one sequence into the other.

Distance Measures



Data Representations

Data Matrix

- For representing n data points/objects with p features/dimensions
- Each row represents a data point
- Each column represents a feature/attribute

$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix}$$

Data Representations

Distance / Proximity Matrix

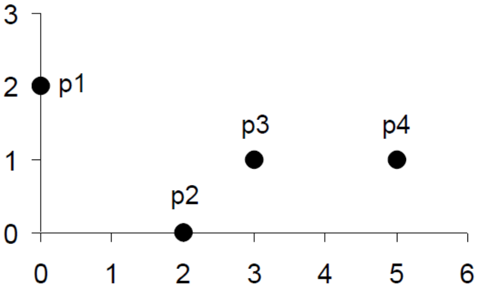
- A square symmetric/triangular matrix
- For representing the distance among the *n* data points
- Each entity represents the distance between the row and column data point

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n,1) & d(n,2) & \ldots & \ldots & 0 \end{bmatrix}$$

Data Representation Example - Problem

Example

- Suppose we have this small dataset. It contains 4 data points and 2 features (*x* and *y*)
- What will be the data matrix?
- What will be the dissimilarity matrix for Manhattan Distance?



Data Representation Example - Solution

point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

Table: Data Matrix

(a) Dissimilarity Matrix for Manhattan Distance

	p1	**p2**	**p3**	**p4**
p1	**0**	**4**	**4**	**6**
p2	4	**0**	**2**	**4**
p3	4	2	**0**	**2**
p4	6	4	2	**0**

(b) Dissimilarity Matrix for Euclidean Distance

	p1	**p2**	**p3**	**p4**
p1	**0**	**2.828**	**3.162**	**5.099**
p2	2.828	**0**	**1.414**	**3.162**
p3	3.162	1.414	**0**	**2**
p4	5.099	3.162	2	**0**

K-Means

K-Means

- Partitional Clustering Algorithm
- Each cluster has a centroid (central point)
- Each point is assigned to the cluster with the closest centroid
- **Number of cluster k** must be known and specified in advance

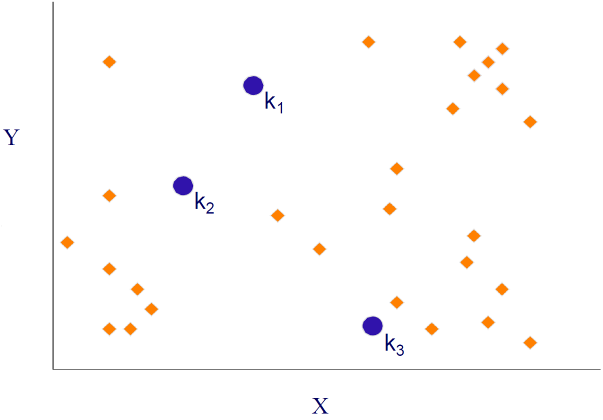
Algorithm 1 k-means algorithm

1: Specify the number k of clusters to assign.
2: Randomly initialize k centroids.
3: **repeat**
4:     **expectation:** Assign each point to its closest centroid.
5:     **maximization:** Compute the new centroid (mean) of each cluster.
6: **until** The centroid positions do not change.

K-Means Example

Step 1:

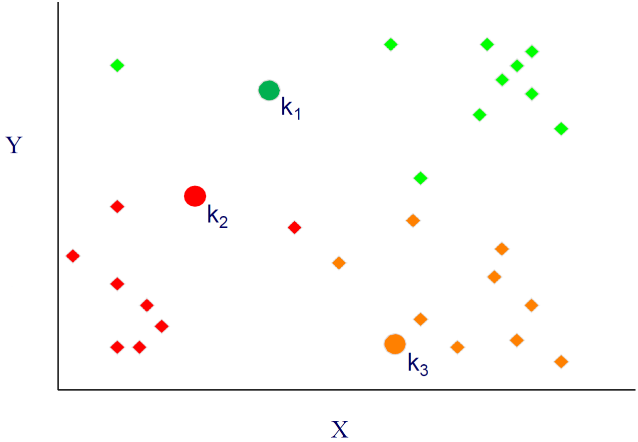
- For k = 3, randomly pick 3 initial centroids



K-Means Example

Step 2:

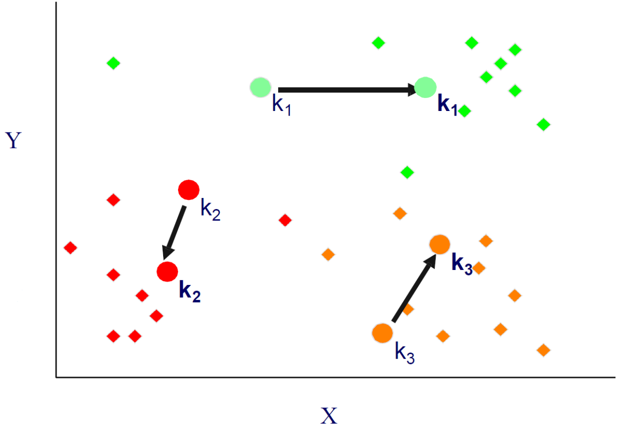
- Assign each point to the closest centroid (using a distance measure)



K-Means Example

Step 3:

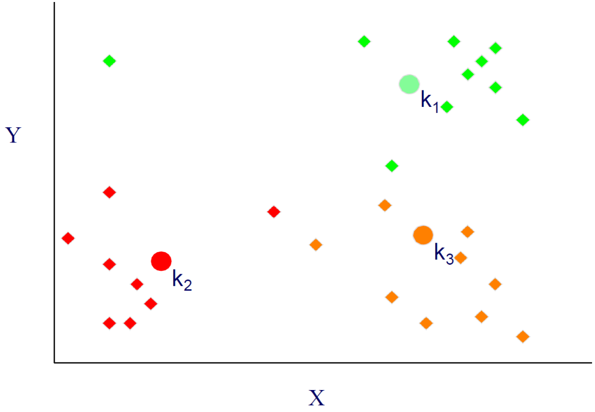
- Move the centroid to the **mean** of each cluster



K-Means Example

Step 4:

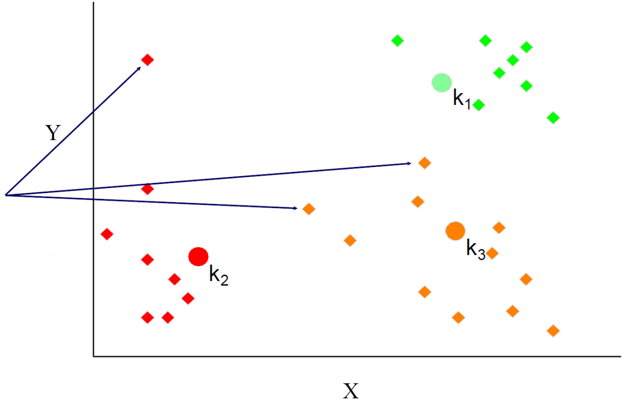
- Reassign points to the suitable cluster, if they are now closer to a different centroid



K-Means Example

Step 4:

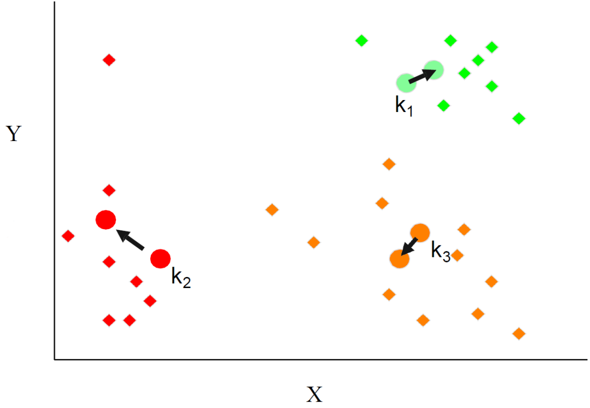
- The reassigned points are:



K-Means Example

Step 5:

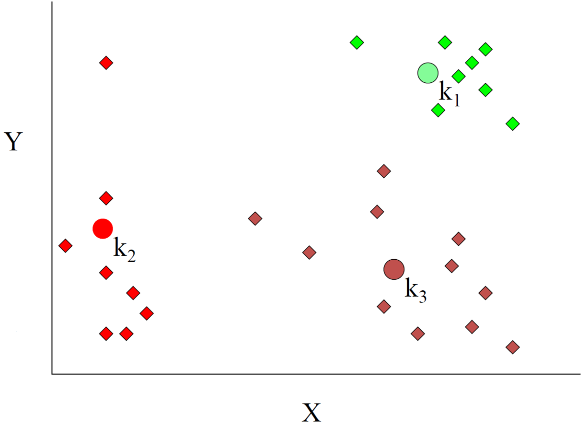
1. Recompute cluster means
2. Move centroids to new cluster means



K-Means Example

Step 5:

- The new centroids (which are the cluster means):



K-Means Convergence

When do we stop? -> Convergence

- no (or minimum) change of centroids
- no (or minimum) reassignments of data points to different clusters
- stopping after a predefined number of iterations
- setting a goal value for an evaluation metric (ex: minimum decrease in the sum of squared errors (SSE))

K-Means Evaluation

Sum of Squared Errors (SSE)

For each point, the error is the distance to the nearest centroid

$$\mathcal{SSE} = \sum_{j=1}^{k} \sum_{x \in C_j} distance(x, m_j)^2$$

where:

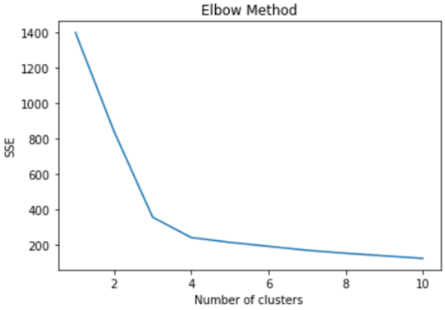
- C_j is the j^{th} cluster
- m_j presents the centroid of C_j
- $distance(x, m_j)$ is the distance between a data point x and the centroid m_j

Given several clusterings (groupings), the one with the smallest SSE is the most preferable

Finding the optimal k

Elbow Method

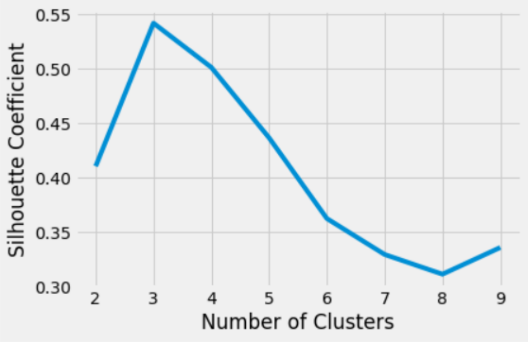
- Elbow method is used to find the optimal number of clusters for a given dataset
- The method works by plotting the SSE as a function of the number of clusters and picking the elbow to be the number of clusters



Finding the optimal k

Graph Based

- Plotting any evaluation metric as a function of the number of clusters
- Choose the number of clusters that optimizes the metric



K-Means in Action - Python - Complete Example

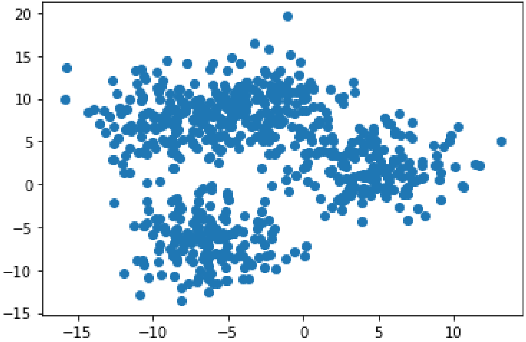
1. Import the needed Libraries

```python
# Import needed libraries
from matplotlib import pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.metrics import davies_bouldin_score
from sklearn.preprocessing import StandardScaler
```

K-Means in Action - Python - Complete Example

2. Create a sample dataset (or upload one)

```
# Creating a sample datasets
X, y = make_blobs(n_samples=700, centers=4, cluster_std=2.75, random_state=42)
plt.scatter(X[:,0], X[:,1])
```



K-Means in Action - Python - Complete Example

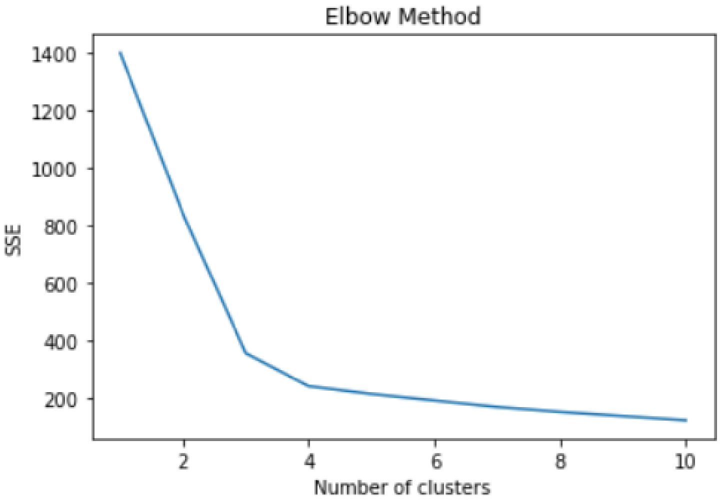
3. Scale using StandardScaler from sklearn

```
#Scale the features
scaler = StandardScaler()
scaled_X = scaler.fit_transform(X)
```

4. Perform the elbow method and choose the optimal k value

```
sse = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i).fit(scaled_X)
    sse.append(kmeans.inertia_)
plt.plot(range(1, 11), sse)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('SSE')
plt.show()
```

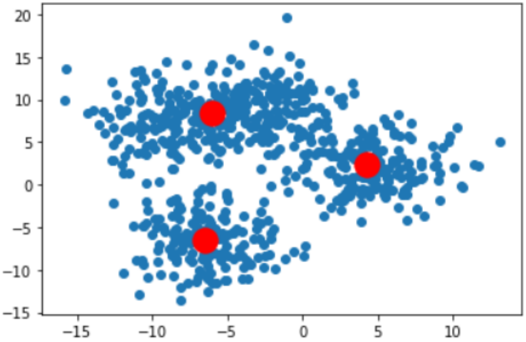
K-Means in Action - Python - Complete Example



K-Means in Action - Python - Complete Example

5 Apply K-means

```
kmeans = KMeans(n_clusters=3, init='k-means++', max_iter=300, n_init=10, random_state=0)
pred_y = kmeans.fit_predict(X)
plt.scatter(X[:,0], X[:,1])
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=300, c='red')
plt.show()
```



K-Means in Action - Python - Complete Example

6. Evaluate the resulting clusters

Evaluate the clustering results - Silhouette

```
silhouette_score = silhouette_score(scaled_X, kmeans.labels_)
```

Evaluate the clustering results - SSE - Inertia

```
sse_score = kmeans.inertia_
```

Evaluate the clustering results - Davies Bouldin Score

```
db_score = davies_bouldin_score(scaled_X, kmeans.labels_)
```

Meanshift Clustering

Meanshift Clustering

- Density-Based Clustering
- In simple words: Shifting to higher density regions by shifting to the mean, in an iterative process
- *Sliding Window* algorithm. A circular sliding window with radius r is used. (the radius is referred to as bandwidth or kernel)
- Density of a sliding window is represented by the number of points inside the window
- Meanshift is a centroid-based algorithm used to find dense areas of data points and locate the center points of each group
- Result of Meanshift: Final set of center points and their corresponding clusters.

A complete example on Meanshift with Code in Python

Meanshift Algorithm

Algorithm

1. Begin with a circular sliding window with a radius r, centered at a random data point.
2. At each step, shift the center of the sliding window to the mean of all points inside the window (thus to regions of higher density)
3. Stop shifting, when we are no longer adding more points to the window (i.e. we are no longer increasing the density in the window). At this point, we have found the center of the future cluster.
4. Steps 1 to 3, are in fact done with multiple sliding windows until all points become in one window:
   - When multiple sliding windows overlap, the one having the greatest number of points is preserved (the most dense window)
   - Data points are then clustered according to the sliding window they reside in.

DBSCAN [1]

Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

- DBSCAN eliminates noise points and returns the clustering of the remaining points
- The Parameters of DBSCAN:
  1. **minPts:** The minimum number of points clustered together for a region to be considered dense
  2. **eps(ϵ):** A distance, used to locate the points in the neighborhood of any point
- Some Concepts in DBSCAN
  1. **Eps-neighborhood of a point p ($N_{Eps}(p)$):** is defined by

$$(N_{Eps}(p) = \{q \in D | dist(p,q) \leq Eps\}$$

[1] *From* A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, Ester et al., 1996

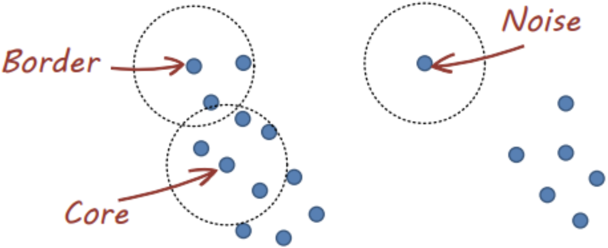
DBSCAN

DBSCAN

- **Density:** Number of points within a specific radius Epsilon (ε)
- Divides data points into 3 types:
  - **Core Point:** A point that has at least a specified number of neighboring points (MinPts) within the specified radius ε
    - The point itself is counted as well
    - These points form the interior of a dense region (cluster)
  - **Border Point:** A point with fewer points than MinPts within ε, but is the neighborhood of a *core point*
  - **Noise Point:** Any point that is neither a core point nor a border point

DBSCAN in Action

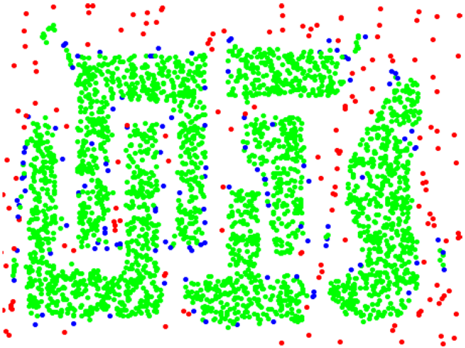
DBSCAN Points Example



$$\varepsilon = 1.0$$
$$MinPts = 5$$

DBSCAN Example

Original Points



Point types: core, border and noise

DBSCAN

Algorithm

1. Pick a random data point *p* as your first point.
2. Mark *p* as visited
3. Extract all points present in its neighborhood (up to *eps* distance from the point), and call it a set **nb**
4. If $nb \geq minPts$, then
   a. Consider *p* as the first point of a new cluster
   b. Consider all points withing *eps* distance (members of *nb*) as other points in this cluster
   c. Repeat step b. for all points in *nb*
5. Else, label *p* as noise
6. Repeat steps 1-5 till the entire dataset is labeled. Thus the clustering is complete.

Density-Based Algorithms in Python

1. **DBSCAN**

```
from sklearn.cluster import DBSCAN
db = DBSCAN(eps=0.4, min_samples=20)
db.fit(X)
```

2. **Meanshift**

```
from sklearn.cluster import MeanShift
mshift = MeanShift().fit(X)
```

Hierarchical Clustering

Recall

Hierarchical clustering produces a set of nested clusters organized as a tree called **dendrogram**

Dendrogram: All what you need to know! (1)

- The core concept of hierarchical clustering lies in the construction and analysis of the resulting dendrogram.
- Tree like structure that shows the sequence of merges or splits applied to the data points.
- The diagram is either constructed in a bottom-up manner (agglomerative algorithm) or in the opposite manner, top-bottom (divisive algorithm).
- Once constructed, the diagram is analyzed by slicing it horizontally.

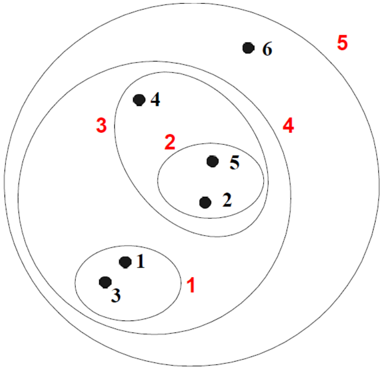
Hierarchical Clustering

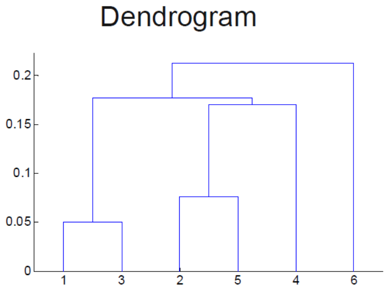
Dendrogram: All that you need to know! (2)

- The core concept of hierarchical clustering lies in the construction and analysis of the resulting dendrogram.
- Tree-like structure that shows the sequence of merges or splits applied to the data points.
- The diagram is either constructed in a bottom-up manner (agglomerative algorithm) or in the opposite manner, top-bottom (divisive algorithm).
- Once constructed, the diagram is analyzed by slicing it horizontally.
- All the possibilities of clusters are provided through the dendrogram
- The final clustering is picked by a horizontal cut through the dendrogram (*search for large gaps to cut* . . .)

Hierarchical Clustering - Dendrogram - Overview

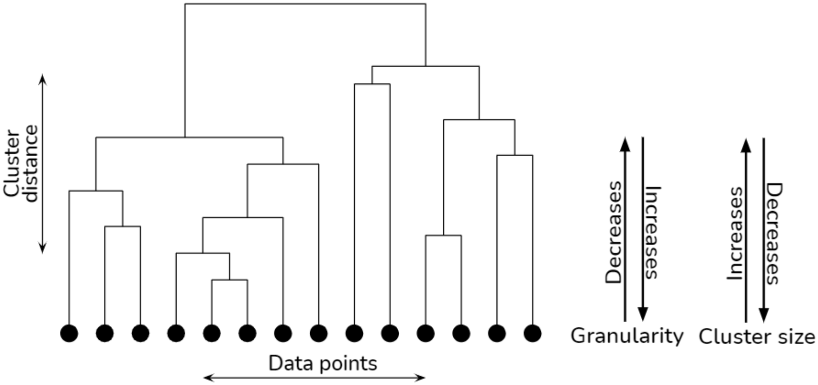
1. Records the sequence of clustering.





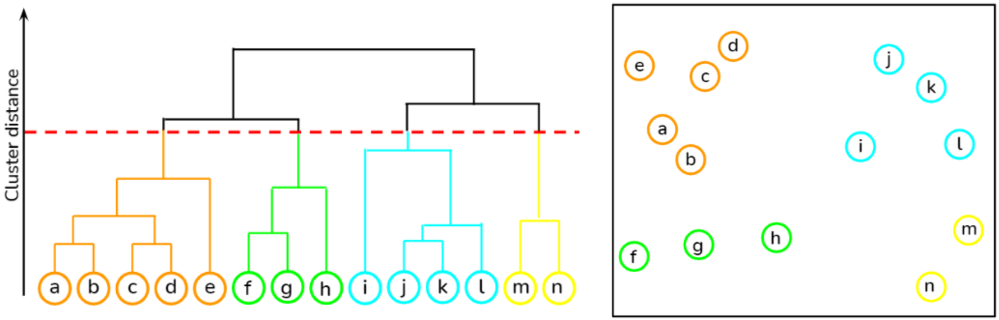
Hierarchical Clustering - Dendrogram - Overview

1. Records the sequence of clustering.



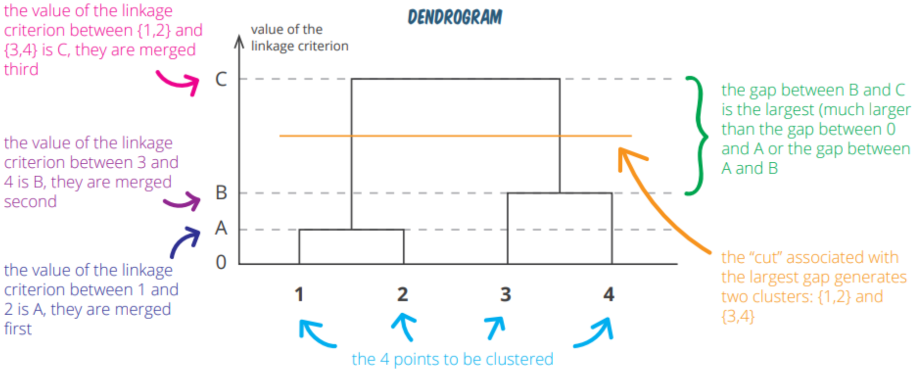
Hierarchical Clustering - Dendrogram - Analysis

2. Analyzed by slicing it horizontally



Hierarchical Clustering - Dendrogram - Analysis

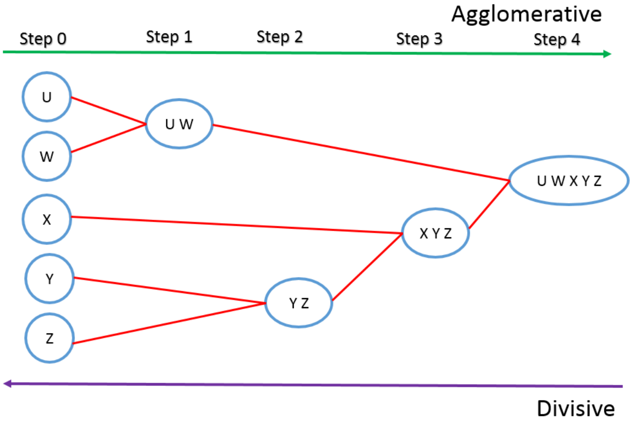
2. Analyzed by slicing it horizontally (search for larger gaps)



Auxiliary Reference

Hierarchical Clustering - Dendrogram - Construction

3. Constructed in a top-bottom manner (divisive) or bottom-up manner (agglomerative)



Construction Algorithms - Agglomerative

Definition

This algorithm starts with the points as individual clusters, and at each step, the closest pairs of clusters are merged until only one final cluster is left

Algorithm

1. Compute the proximity matrix
2. Let each data point be a cluster
3. **Repeat**
   1. Merge the two closest clusters
   2. Update the proximity matrix (*But how? Measuring proximity between clusters?*)

   Unitl only a single cluster is left

The key operation and additional step here is the computation of the proximity between two clusters

Agglomerative - Cluster Distance Measures

Linkage Criteria

- The linkage criteria refers to how the distance between clusters is measured
- The distance between two clusters, In:
  1. **Single Linkage**: is the **shortest** distance between an element in one cluster and an element in the other
  2. **Complete Linkage**: is the **longest** distance between an element in one cluster and an element in the other
  3. **Average Linkage**: is the **average** distance between each point in one cluster to every point in the other cluster. This compromises between single and complete linkage, as it is less sensitive to noise and outliers than single linkage.
  4. **Ward Linkage**: is the **sum of squared differences** within all clusters

Linkage Criteria

Single Linkage

$$\mathrm{l}(A, B) = \min \{\mathrm{d}(a, b) : a \in A, b \in B\}$$

Complete Linkage

$$\mathrm{l}(A, B) = \max \{\mathrm{d}(a, b) : a \in A, b \in B\}$$

Average Linkage

$$\mathrm{l}(A, B) = \frac{1}{|A| \cdot |B|} \sum_{a \in A} \sum_{b \in B} \mathrm{d}(a, b)$$

Linkage Criteria: Ward Criterion

Ward Linkage

- Ward Criterion defines the distance between 2 clusters A and B as how much the sum of squares will increase when we **merge** them.
- Ward tries to minimize Δ as it moves forward in clustering

$$\Delta(A,B) = \sum_{x \in A \cup B} \|x - m_{A \cup B}\|^2 - \sum_{a \in A} \|a - m_A\|^2 - \sum_{b \in B} \|b - m_B\|^2$$

- *where*
  - m_x is the center of cluster x
- Ward is known to be used with Euclidean Distance

Helper Note: To simplify the equation, it is the intra-cluster of the merged cluster minus the intra-cluster of the first cluster minus the intra-cluster of the second cluster*

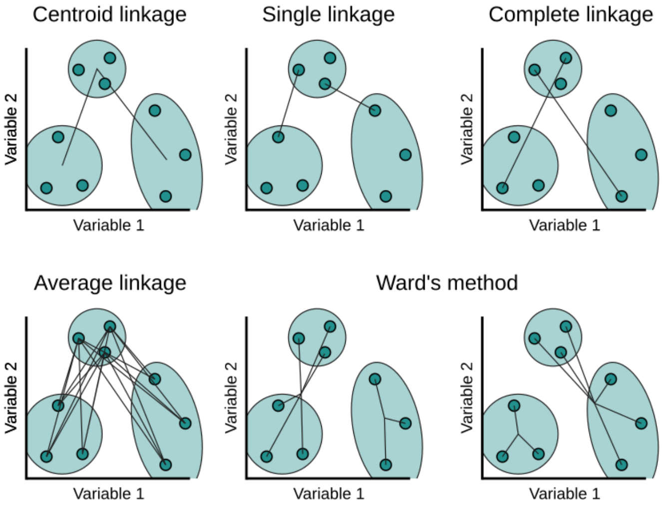
Linkage Criteria: Ward Criterion

Simpler equation for Ward's Method

$$\Delta(A,B) = \sum_{x \in A \cup B} \|x - m_{A \cup B}\|^2 - \sum_{a \in A} \|a - m_A\|^2 - \sum_{b \in B} \|b - m_B\|^2$$
$$= \frac{n_A . n_B}{n_A + n_B} \|m_A - m_B\|^2$$

- *where:*
  - m_x is the center of cluster x
  - n_x is the number of points in cluster x

Linkage Criteria: A visual glimpse



Example - Applying Linkage Criteria

Example

Given the following dataset with 5 points, and one feature.

	a	b	c	d	e
f	1	2	4	5	6

Consider we have 2 clusters: $C_1 = \{a, b\}$ and $C_2 = \{c, d, e\}$

1. Draw and fill the proximity matrix of the provided dataset using Euclidean distance
2. Calculate the 4 different cluster distances between C_1 and C_2 (single, complete, average, ward) using Euclidean distance

Example - Solution

The proximity matrix:

	a	b	c	d	e
a	0	1	3	4	5
b	1	0	2	3	4
c	3	2	0	1	2
d	4	3	1	0	1
e	5	4	2	1	0

Single Linkage

$$\operatorname{dist}(C_1, C_2) = \min\{(a,c),(a,d),(a,e),(b,c),(b,d),(b,e)\}$$
$$= \min\{3,4,5,2,3,4\} = 2$$

Example - Solution

Complete Linkage

$$\begin{aligned}\mathrm{dist}\,(C_1, C_2) &= \max\{(a,c),(a,d),(a,e),(b,c),(b,d),(b,e)\}\\ &= \max\{3,4,5,2,3,4\} = 5\end{aligned}$$

Average Linkage

$$\begin{aligned}\mathrm{dist}\,(C_1, C_2) &= \frac{d(a,c),d(a,d),d(a,e),d(b,c),d(b,d),d(b,e)}{n_1 \times n_2}\\ &= \frac{3+4+5+2+3+4}{2 \times 3}\\ &= \frac{21}{6} = 3.5\end{aligned}$$

Example - Solution

Ward Linkage

$$\Delta(C_1, C_2) = \frac{n_{C_1} . n_{C_2}}{n_{C_1} + n_{C_2}} \left\| m_{C_1} - m_{C_2} \right\|^2$$
$$= \frac{6}{5} \left\| 1.5 - 5 \right\|^2$$
$$= \frac{6}{5} \left\| -3.5 \right\|^2$$
$$= \frac{6}{5} \times 12.25 = 14.7$$

Hierarchical Agglomerative Clustering in Python - Complete Example

Dataset

For this example, we will use the [Wholesale customer data.csv]

1. Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

Hierarchical Agglomerative Clustering in Python - Complete Example

2. Load and Visualize your data

```
data = pd.read_csv("Wholesale customers data.csv")
data.head()
```

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	2	3	12669	9656	7561	214	2674	1338
1	2	3	7057	9810	9568	1762	3293	1776
2	2	3	6353	8808	7684	2405	3516	7844
3	1	3	13265	1196	4221	6404	507	1788
4	2	3	22615	5410	7198	3915	1777	5185

Hierarchical Agglomerative Clustering in Python - Complete Example

3. Normalize your data - preprocessing step

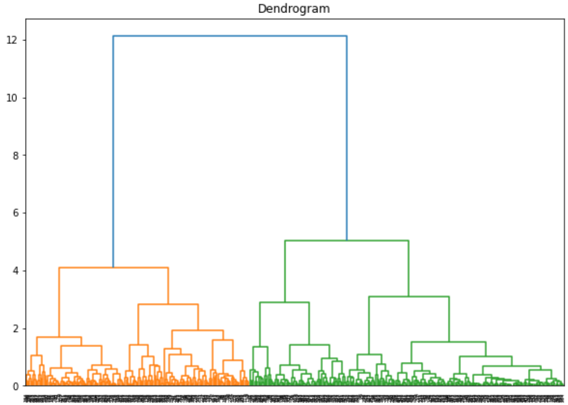
```
from sklearn.preprocessing import normalize
scaled_data = normalize(data)
scaled_data = pd.DataFrame(scaled_data, columns=data.columns)
scaled_data.head()
```

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	0.000112	0.000168	0.708333	0.539874	0.422741	0.011965	0.149505	0.074809
1	0.000125	0.000188	0.442198	0.614704	0.599540	0.110409	0.206342	0.111286
2	0.000125	0.000187	0.396552	0.549792	0.479632	0.150119	0.219467	0.489619
3	0.000065	0.000194	0.856837	0.077254	0.272650	0.413659	0.032749	0.115494
4	0.000079	0.000119	0.895416	0.214203	0.284997	0.155010	0.070358	0.205294

4. Find the optimal number of clusters using dendrogram (horizontal cut at the largest distance)

Hierarchical Agglomerative Clustering in Python - Complete Example

```
import scipy.cluster.hierarchy as sch
plt.figure(figsize=(10, 7))
plt.title("Dendrogram")
dend = sch.dendrogram(sch.linkage(scaled_data, method='ward'))
```



Hierarchical Agglomerative Clustering in Python - Complete Example

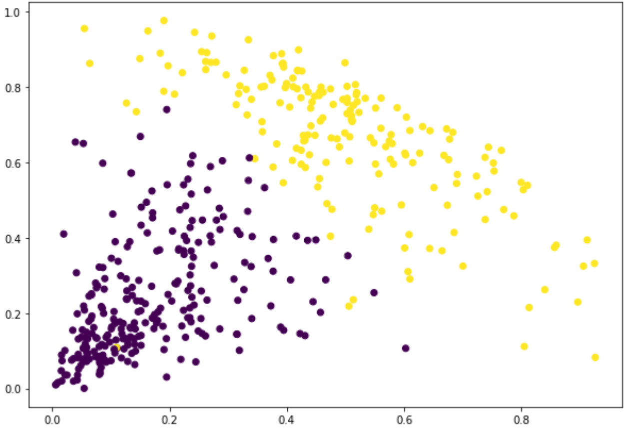
5. Apply Agglomerative Clustering with the optimal value (2 in this case)

```
from sklearn.cluster import AgglomerativeClustering
cluster = AgglomerativeClustering(n_clusters=2, affinity='euclidean', linkage='ward')
cluster.fit_predict(scaled_data)
```

6. Visualize the resulting clusters

```
plt.figure(figsize=(10, 7))
plt.scatter(scaled_data['Milk'], scaled_data['Grocery'], c=cluster.labels_)
```

Hierarchical Agglomerative Clustering in Python - Complete Example



Hierarchical Agglomerative Clustering in Python - Complete Example

7. Evaluate your clustering results

```
from sklearn.metrics import davies_bouldin_score
db_score = davies_bouldin_score(scaled_data, cluster.labels_)
db_score
```

```
0.8166647229022314
```

Clustering Evaluation

Evaluation Metrics

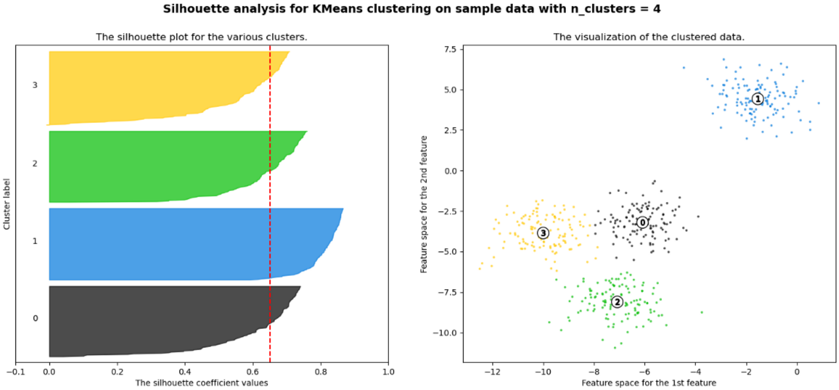
1. **Sum of Squared Errors (SSE)**
2. **Silhouette Score**
3. **Davies Bouldin Index**

Clustering Evaluation

Evaluation Metrics

2. **Silhouette Score** :
   - A metric used to evaluate the goodness of clustering (and to find the optimal number of clusters)
   - Silhouette Score = $\frac{(n-i)}{max(i,n)}$; where:
     - n: the mean distance between a sample and all other points in the next nearest cluster
     - i: the mean distance between a sample and all other points in the same cluster
   - The silhouette score for a set of samples is the mean of the silhouette scores for each sample.
   - Range [-1,1] -> ***How to interpret Silhouette Score?***
     - Closer to 1: Clusters are clearly distinguished and well apart
     - Closer to 0: Distance between clusters is not significant
     - Closer to -1: Clusters are assigned in the wrong way (incorrect clustering)

Silhouette Score Example



Clustering Evaluation

Evaluation Metrics

3. **Davies Bouldin Index**
   - Introduced by David L. Davies and Donald W. Bouldin in 1979
   - This index captures if the clusters are well spaced from each other and if the data points in the clusters are dense enough
   - Defined as the average similarity measure of each cluster with its most similar cluster.
     Similarity is the ratio of within-cluster (intra-cluster) distances to between-cluster (inter-cluster) distances

Clustering Evaluation

Evaluation Metrics

3. **Davies Bouldin Index**
   - Range is [0,∞]. The smaller the value of this index, the better is the clustering.
   - The index is calculated as follows:

$$DB = \frac{1}{n} \sum_{i=1}^{n} \max_{j \neq i} \left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right)$$

   where:

   - n n is the number of clusters
   - σ_i is the average distance of all points in cluster i from the cluster centroid c_i

Evaluation Metrics in Python

Imported Libraries

```
from sklearn.metrics import silhouette_score
from sklearn.metrics import davies_bouldin_score
```

Evaluate the clustering results - Silhouette

```
silhouette_score = silhouette_score(scaled_X, kmeans.labels_)
```

Evaluate the clustering results - SSE - Inertia

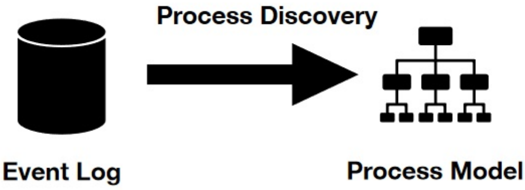
```
sse_score = kmeans.inertia_
```

Evaluate the clustering results - Davies Bouldin Score

```
db_score = davies_bouldin_score(scaled_X, kmeans.labels_)
```

Outline

Context: What about Process Mining ?



Terminology

- **Event log**: an event log $L = \{t_1, t_2, ..., t_k\}$ is a set of k **traces**
- **Trace**: each trace t_i $(1 \leq i \leq k)$ is a set of n_i consecutive events $t_i =< e_{i1}, e_{i2}, ...e_{in_i} >$ made by the same user.
- **Event**: an event e is an **activity** performed by the user of the information system. Each event is characterized by its **frequency** f_e which is the number of times it occurs in L.

Users' traces examples

CaseId	User	Timestamp	Activity
1	$user_1$	2016-01-12T10:34:25	home index
1	$user_1$	2016-01-12T10:34:27	home languages
1	$user_1$	2016-01-12T10:34:28	language selection
1	$user_1$	2016-01-12T10:34:31	catalog show
2	$user_2$	2016-01-12T10:34:26	home index
2	$user_2$	2016-01-12T10:34:29	home periods
2	$user_2$	2016-01-12T10:34:30	catalog show

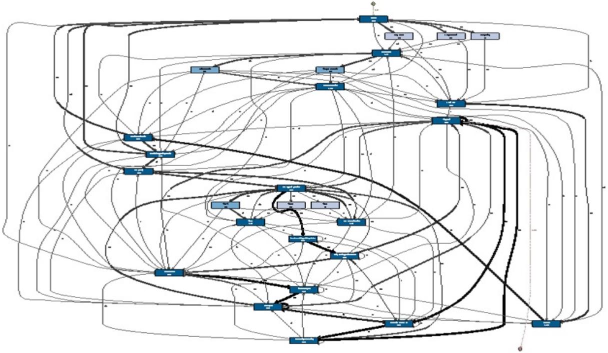
$\boxed{user_1}$ $\boxed{\text{home-index}} \rightarrow \boxed{\text{home-languages}} \rightarrow \boxed{\text{language-selection}} \rightarrow \boxed{\text{catalog-show}}$

$\boxed{user_2}$ $\boxed{\text{home-index}} \rightarrow \boxed{\text{home-periods}} \rightarrow \boxed{\text{catalog-show}}$

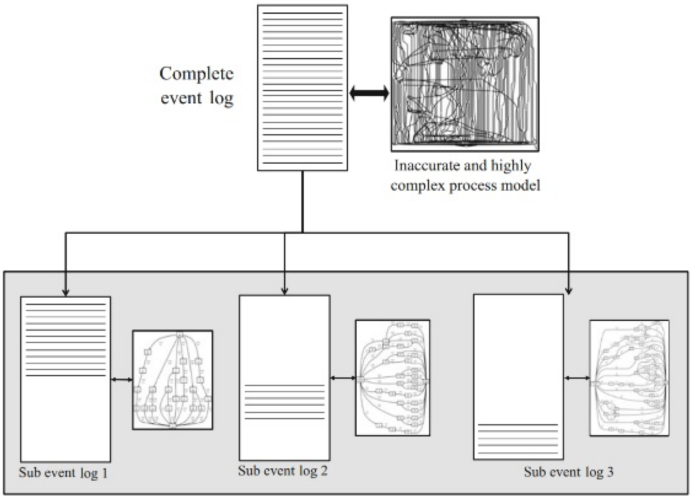
Context :Process Mining and spaghetti models

Spaghetti models ?

- Huge amount of data —> Process Mining techniques will discover complex users' behaviors models.



Context : clustering before modeling



Why trace clustering ?

- Trace clustering has been used as a method to partition event logs in a way that more homogeneous sublogs are obtained, with the hope that process discovery techniques will perform better on the sublogs than if applied to the original log.
- Existing PM techniques perform well on structured processes
- Processes for each users types (novice users, professional users...) or research tasks.
- Process enhancement (by proposing different types of processes for users).
- **How can we identify groups of behaviorally similar traces in an event log?**

Clustering users traces

Trace-based clustering

Similarity between two traces can be measured using the syntax similarity.

Feature-based clustering

Converting each trace into a vector of features based on defined characteristics.

Model-based clustering

Process models are considered as input for the clustering in order to structure traces.

- **Hybrid based clustering** : combines the previous methods.

Trace-based clustering

- **Trace-based clustering**, is the first category that cluster the traces using the syntax similarity.
- It is inspired from the **Levenshtein** distance between two strings.
- A trace can be edited into another trace by substituting, adding or removing events.
- The edit distance between two traces is the minimum number of edit operations required to transform one trace to the second. Less the edit distance is, more the traces are similar.

Trace-based clustering : example

- For two traces or sequences t_1 et t_2, the following edit operations are considered on the activities $A \cup \{-\}$ where $-$ denotes a *gap*. For $a, b \in A$, the pair
- (a, a) denotes a match of activities between t_1 and t_2 at some position $t_1(i)$ and $t_2(j)$. A match can be considered as a substitution of an activities with itself.
- $(a, -)$ denotes the deletion of a in t_1 at some position $t_1(i)$
- $(-, b)$ denotes the insertion b in $t_1(i)$
- (a, b) denotes the replacement/substitution of a in t_1 with b at some position $t_1(i)$ where $a \neq b$

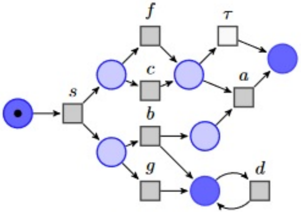
Trace-based clustering: state of the art

- **Bose, R. J. C. and Van der Aalst, W. M. (2009 a), Context aware trace clustering : Towards improving process mining results, in "Proceedings of the 2009 SIAM International Conference on Data Mining", SIAM, 401–412**
- They propose a context-aware approach to trace clustering based on generic edit distance.
- They tackle the sensitivity of the cost function of edit operations in the process.
- They determined the cost by taking into account the context of an event within a trace.

Trace-based clustering: state of the art

- **Chatain, T., Carmona, J. and Van Dongen, B. (2017), Alignment-based trace clustering, in "International Conference on Conceptual Modeling", Springer, 295–308.**
- The clustering approach of this paper assumes an additional input: a process model that describes the current process
- The idea of their algorithm is to group log traces according to their closeness to representative full runs of a given model. Those representative full runs act as **centroids** for the clusters.
- This way, even in case of deviations, incomplete or noisy traces, or even drifts in the process model, a process explanation of the traces in each cluster is available, so that stakeholders can relate them more reliably to the underlying process.

Trace-based clustering: state of the art (Chatain et al., example)



(a) Petri Net.

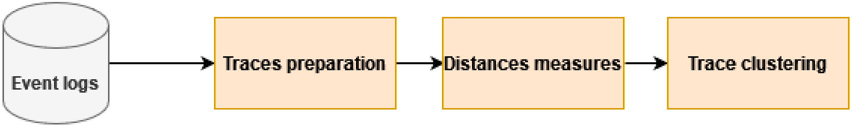
$\langle s, c, g \rangle$
$\langle s, c, g, d \rangle$
$\langle s, f, b, a \rangle$
$\langle s, f, f, a \rangle$
$\langle s, b, f, a \rangle$
$\langle s, g, f, d, d \rangle$
$\langle s, g, f, d, d, d, d \rangle$
$\langle g, c, f, s, d, d \rangle$
$\langle s, d, d, d \rangle$

(b) Log L_1

Centroids	Traces	Distance
$\langle s, c, \tau, g \rangle$	$\langle s, c, g \rangle$	0
	$\langle s, c, g, d \rangle$	1
$\langle s, b, f, a \rangle$	$\langle s, b, f, a \rangle$	0
	$\langle s, f, f, a \rangle$	1
$\langle s, f, b, a \rangle$	$\langle s, f, b, a \rangle$	0
$\langle s, g, f, \tau, d, d \rangle$	$\langle s, g, f, d, d \rangle$	0
$\langle s, g, f, \tau, d, d, d, d \rangle$	$\langle s, g, f, d, d, d, d \rangle$	0
non-clustered	$\langle g, c, f, s, d, d \rangle$	NA
	$\langle s, d, d, d \rangle$	NA

(c) Clusters

Feature-based clustering



- **Feature-based clustering**, is the second category that consists in converting each trace into a vector of features based on defined characteristics before the clustering.
- Various distance metrics in data mining are reused to estimate the similarity between the corresponding traces vectors.
- Subsequently, distance-based clustering algorithms are deployed, such as k-means or agglomerative hierarchical clustering algorithms.

Feature-based clustering: state of the art

- **Song, M., Günther, C. W. and Van der Aalst, W. M. (2008), Trace clustering in process mining, in "International Conference on Business Process Management", Springer, 109–120.**
- The paper presents an approach based on **log profiles**, using trace clustering, i.e., the event log is split into homogeneous subsets and for each subset a process model is created.
- Each trace is transformed into a vector of features based on, for example, the **frequency of activities**, the **frequency of directly-followed relations**, the resources involved, etc.

Feature-based clustering: state of the art (Song et al., example)

Case ID	log events
1	(A,John),(B,Mike),(D,Sue),(E,Pete),(F,Mike),(G,Jane),(I,Sue)
2	(A,John),(B,Fred),(C,John),(D,Clare),(E,Robert),(G,Mona),(I,Clare)
3	(A,John),(B,Pete),(D,Sue),(E,Mike),(F,Pete),(G,Jane),(I,Sue)
4	(A,John),(C,John),(B,Fred),(D,Clare),(H,Clare),(I,Clare)
5	(A,John),(C,John),(B,Robert),(D,Clare),(E,Fred),(G,Robert),(I,Clare)
6	(A,John),(B,Mike),(D,Sue),(H,Sue),(I,Sue)

Table 1. Example process logs (A: Receive a item and repair request, B: Check the item, C: Check the warranty, D: Notify the customer, E: Repair the item, F: Test the repaired product, G: Issue payment, H: Send the cancellation letter, I: Return the item)

Case ID	Activity Profile									Originator Profile								
	A	B	C	D	E	F	G	H	I	John	Mike	Sue	Pete	Jane	Fred	Clare	Robert	Mona
1	1	1	0	1	1	1	1	0	1	1	2	2	1	1	0	0	0	0
2	1	1	1	1	1	0	1	0	1	2	0	0	0	0	1	2	1	1
3	1	1	0	1	1	1	1	0	1	1	1	2	2	1	0	0	0	0
4	1	1	1	1	0	0	0	1	1	2	0	0	0	0	1	3	0	0
5	1	1	1	1	1	0	1	1	0	2	0	0	0	0	1	2	2	0
6	1	1	0	1	0	0	0	1	1	1	1	3	0	0	0	0	0	0

Feature-based clustering: state of the art

- **Bose, R. J. C. and van der Aalst, W. M. (2009 b), Trace clustering based on conserved patterns: Towards achieving better process models, in "International Conference on Business Process Management", Springer, 170–181.**
- The basic idea is to consider k-gram of activities that are conserved across multiple traces (variable k).
- Finding similar regions (sequence of activities) common within a trace and/or across a set of traces in an event log signifies some set of common functionality accessed by the process.
- The observed k-grams are the different patterns such as the **Maximal Repeat Set**, as well as the **Super Maximal Repeat Set** and the **Near Super Maximal Repeats Set** to constitute the vector of a particular trace.

Feature-based clustering: state of the art (bose et al., example)

- **Maximal Repeat**: A maximal repeat in a sequence, T, is defined as a subsequence α that occurs in a maximal pair in T.
- **Super Maximal Repeat**: A super maximal repeat in a sequence is defined as a maximal repeat that never occurs as a substring of any other maximal repeat.
- **Near Super Maximal Repeat**: A maximal repeat α is said to be a near super maximal repeat if and only if there exists at least one instance of α at some location in the sequence where it is not contained in another maximal repeat

Id	Trace	Maximal Repeat Set	Super Maximal Repeat Set	Near Super Maximal Repeat Set
T_1	aabcdbbcda	{a, b, bcd}	{a, bcd}	{a, b, bcd}
T_2	dabcdabcbb	{b, dabc}	{dabc}	{b, dabc}
T_3	bbbcdbbbccaa	{a, b, c, bb, bbbc}	{a, bbbc}	{a, c, bbbc}
T_4	aaadabbccc	{a, b, c, aa, cc}	{b, aa, cc}	{a, b, aa, cc}
T_5	aaacdcdcbedbcc-badbdebdc	{a, b, c,d, e, aa, bd, cb, db, dc, cdc}	{e, aa, bd, cb, db, cdc}	{a, c, e, aa, bd, cb, db, dc, cdc}

Feature-based clustering : state of the art (L3I research works)

- **Trabelsi, M., Suire, C., Morcos, J. and Champagnat, R. (2021 a), A new methodology to bring out typical users interactions in digital libraries, in "2021 ACM/IEEE Joint Conference on Digital Libraries (JCDL)", 11–20.**
- Frequent Sub-Sequences (FSS) in the traces can contribute to distinguish users and tasks.
- Grouping the traces based on the frequent sub-sequences (FSS).
- An $FSS = < e_1, ..., e_n >$ contains a finite set of events e of length n (n > 1) where their events are executed in the order at least two time.
- Converting traces using a particular (FSS) encoding.
- Each identified (FSS) in a trace is replaced by its encoding.

Feature-based clustering: state of the art (Trabelsi et al., L3I research works)

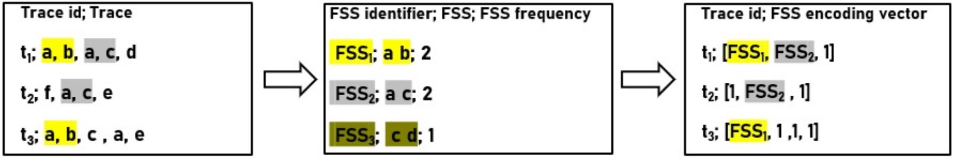
- The FSS encoding itself has to consider many factors to effectively distinguish traces from different clusters.
  - **The length of the FSS**: is the number of events in the FSS.
  - **The frequency of the FSS**: is the number of times the FSS occurs in the whole event logs. The FSS with the highest frequency f_{FSS} is important.
  - **The frequency of events in the FSS**: The difference between two FSS with same frequency and length is underlined by the frequency of their events.
  - **The direct succession relation between events in the FSS**: The encoding takes into consideration the frequency of direct relations between events in the FSS.

Feature-based clustering: state of the art (Trabelsi et al., L3I research works)

$$\text{Encoding (FSS)} = \frac{1}{f_{FSS} \sum_{i=1}^{n-1} f_{e_i} f_{e_{i+1}} f_{r_{i,i+1}}}$$

- f_{FSS} is the frequency of the FSS
- n is its length
- f_{e_i} is the frequency of the event
- $f_{r_{i,i+1}}$ is the frequency of the direct relation between events
- FSS Encoding value $\in [0, 1]$

Feature-based clustering: state of the art (L3I research works, Trabelsi et al., 2021)



- The Prefixspan[2] algorithm was used to extract the sequential patterns *FSS*.
- The extracted *FSS* with a different length *n* are sorted at first according to their lengths and secondly according to their frequencies f_{FSS}

[2] https://pypi.org/project/prefixspan/

Feature-based clustering: state of the art (Trabelsi et al., L3I research works)

Algorithm 1: The FSS Encoding algorithm

Data: Original logs file R
Result: Logs Files F generated according the found clusters
begin
- Convert the original event logs R to sequence of traces L;
- From L, find frequent sub-sequences FSS based on defined features;
- For each element in L, find the most frequent FSS and replace found FSS by their encoding;
- Remove elements in L where no FSS found;
- For each element in L, Gaps between FSS encoding will be replaced by "1";
- Do distance measurement between trace vectors L ;
- Do clustering of L;
- Generate Logs Files F according to the found clusters;
- Return F;

Model-based clustering

- **Model-based clustering**, is the third category that assumes that accurate models are discovered from homogeneous sub-logs.
- The focus is directly on the quality of discovered models and the distribution of traces among clusters.
- The process model is considered as input for the clustering in order to structure traces. These traces are used back to mine process models.
- The obtained clusters strongly depend on the conformance-checking measures used for evaluating the accuracy of discovered process models.

Model-based clustering: state of the art

- **Veiga, G. M. and Ferreira, D. R. (2009), Understanding spaghetti models with sequence clustering for prom, in "International conference on business process management", Springer, 92–103.**
- Authors combined trace clustering with First order Markov models using a hierarchical approach.
- Initially, random clusters are built, and traces are distributed among them. Consequently, the cluster models (state transition probabilities of the Markov chain of each cluster) are evaluated.
- Then iteratively, traces are re-assigned to the clusters and evaluation is done again until the algorithm converges, and cluster models do not change.

Model-based clustering: state of the art

- **De Weerdt, J., Vanden Broucke, S., Vanthienen, J. and Baesens, B. (2013), "Active trace clustering for improved process discovery", IEEE Transactions on Knowledge and Data Engineering 25(12), 2708–2720**
- Authors tried to find the optimal distribution of traces between clusters that leads to maximum quality of process models of clusters.
- They do not aim to find the similarity between traces, but rather they cluster traces that fit in a certain process model.
- A new approach based on active learning that first takes unique cases and, based on their distance or frequency, they are clustered together as primal clusters.
- Clusters accept members only if the fitness is optimized, otherwise traces are allocated to a thrash cluster or are distributed equally between other clusters.

Hybrid clustering

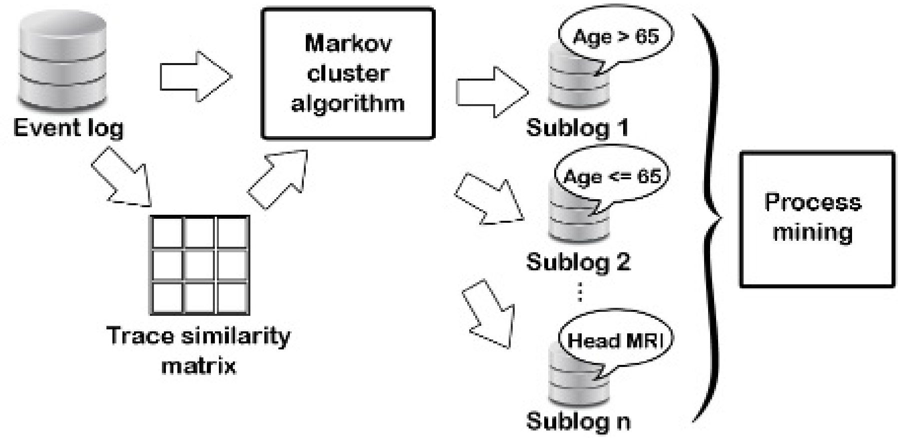
- **Hybrid clustering**, is the last category…
- Combining existing trace clustering categories.

Hybrid clustering: state of the art

- **Hompes, B., Buijs, J., Van der Aalst, W., Dixit, P. and Buurman, J. (2015), Discovering deviating cases and process variants using trace clustering, in "Proceedings of the 27th Benelux Conference on Artificial Intelligence (BNAIC), November", 5–6.**
- Combines the model-based and feature vector-based approaches.
- Traces are transformed into vectors using a trace profiling approach and a similarity matrix is calculated by applying the Cosine similarity measure.
- Eventually, similarity matrix is the input of the MCL algorithm[3] (Markov Cluster Algorithm).
- The graph clustering algorithm is able to find variations and deviations of a process based on a set of selected perspectives.

[3] https://towardsdatascience.com/markov-clustering-algorithm-577168dad475

Hybrid clustering: state of the art (Hompes et al., 2015)



Hybrid clustering: state of the art

- **De Koninck, Pieter, and Jochen De Weerdt. "Scalable mixed-paradigm trace clustering using super-instances." 2019 International Conference on Process Mining (ICPM). IEEE, 2019.**
- General idea:
  - Combine the strengths of the two most prominent trace clustering paradigms (Trace similarity-driven (or distance-driven) techniques and Model-driven techniques)
- Two-step approach:
  - Learn super-instances using a simple distance-driven clustering (e.g. k-means)
  - Apply a model-driven clustering technique to the super-instances to obtain a final clustering

Hybrid clustering : state of the art (De Koninck et al., 2019)

Algorithm 1 Algorithm TraCluSI (Trace Clustering using Super-Instances)

Input: G := a Grouped Event Log, k:= the desired number of clusters
Input: Configuration: n:= the number of super-instances, $Strategy$:= a super-instance selection strategy, PD:= a process discovery technique, m:= a process model quality metric , ctv:= clustering threshold for the quality metric
Output: $\{C_j\}_{j=1}^{k}$:= An ordered set of clusters

$\{Super_i\}_{i=1}^{n}$:= $\emptyset$ % Initialize super-instances empty
$\{Sub_i\}_{i=1}^{n}$:= $\emptyset$ % Initialize sub-instances empty

Phase 1

X := Featurize(G) % X is a clusterable dataset representing the traces
$\{O_i\}_{i=1}^{n}$:= K-Means(X, n) % O is an overclustering of X

Phase 2

for i := $(1 \rightarrow |n|)$ **do** % Assign super-instance for each cluster in the overclustering
  if $Strategy$='Frequency' **then**
    $Super_i$:= Most frequent distinct process instance in O_i
    Sub_i := Set of all other distinct process instances
  else if $Strategy$='Centrality' **then**
    $Super_i$:= Most central distinct process instance in O_i % Closest distance to cluster centroid
    Sub_i := Set of all other distinct process instances
  end if
end for

Phase 3

$\{SC_j\}_{j=1}^{k}$:= ActiTraC(S, k, PD, m, ctv) % Cluster the super-instances in an active manner
for j := $(1 \rightarrow |k|)$ **do** % For each cluster
  for i := $(1 \rightarrow |SC_j|)$ **do** % For each super-instance
    $\{C_j\}$:= $C_j \cup Super_{SC_i} \cup Sub_{SC_i}$ % Add the super-instance and its corresponding sub-instances to the final clusters
  end for
end for
return C

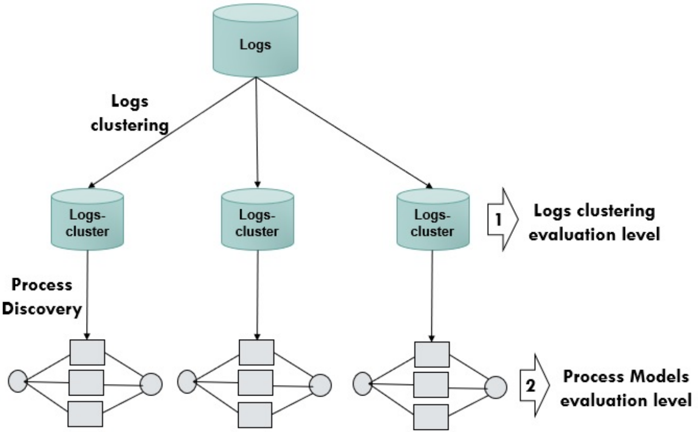
State of the art: summary

Summary

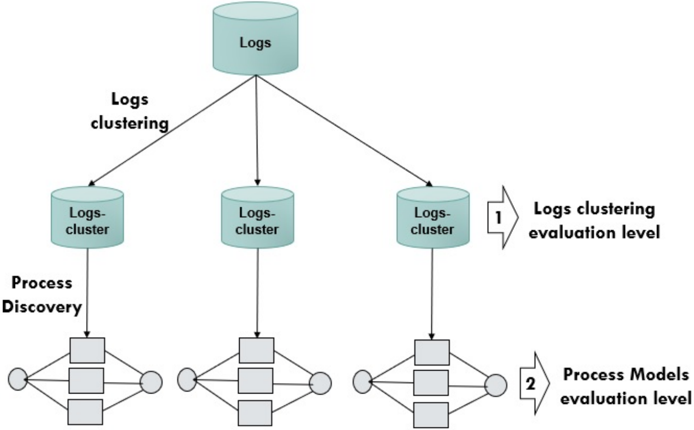
Trace-based **Feature-based** **Model-based** **Hybrid**

Method	Dataset	Traces processing	Clustering
bose et al., 2009	Telephone repair	Edit distance	Hierarchical clustering
Di Francescomarino et al.,2016	Healthcare	Edit distance	DBSCAN
Chatain et al., 2017	Synthetic	Edit distance	Closeness-centroids
Song et al., 2008	Healthcare	Frequent features	Multiple algorithms
bose et al., 2009	Healthcare	n-gram	Hierarchical clustering
Ceravolo et al.,2017	Industry	Frequent features	Multiple algorithms
Trabelsi et al., 2021	Digital Libraries	Frequent subsequences	DBSCAN/Meanshift
Veiga et al., 2009	Administration	Marcov chains	Hierarchical clustering
De Weerdt et al., 2013	Insurance	Active learning	k-clusters
Hompes et al., 2015	Healthcare	Cosine distance	Markov algorithm
De Koninck et al., 2019	Municipality	Frequent features	Active learning

Evaluation levels



Evaluation levels



Clustering evaluation measures

- **Silhouette**
- **Davies-Bouldin**

Process evaluation measures

- **Fitness**
- **Precision**
- **Generalization**
- **F-measure**

Outline

Information system example: Gallica

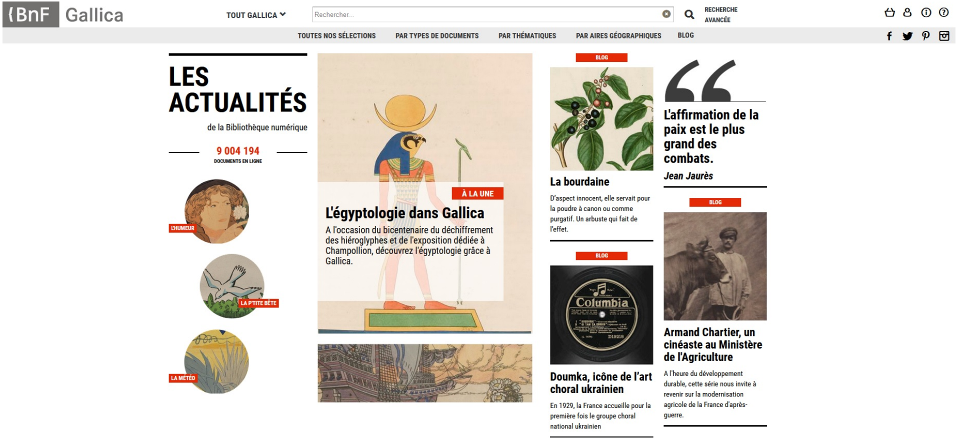


Image from `https://gallica.bnf.fr`

User's Journey Modeling in Gallica's Digital library, Trabelsi et al. 2022

Process Mining for modeling Digital Library users' behaviors

Is process mining appropriate to extract knowledge from DL users' journeys?

Digital Library Logs Transformation

How to transform real logs into logs compatible with process mining techniques?

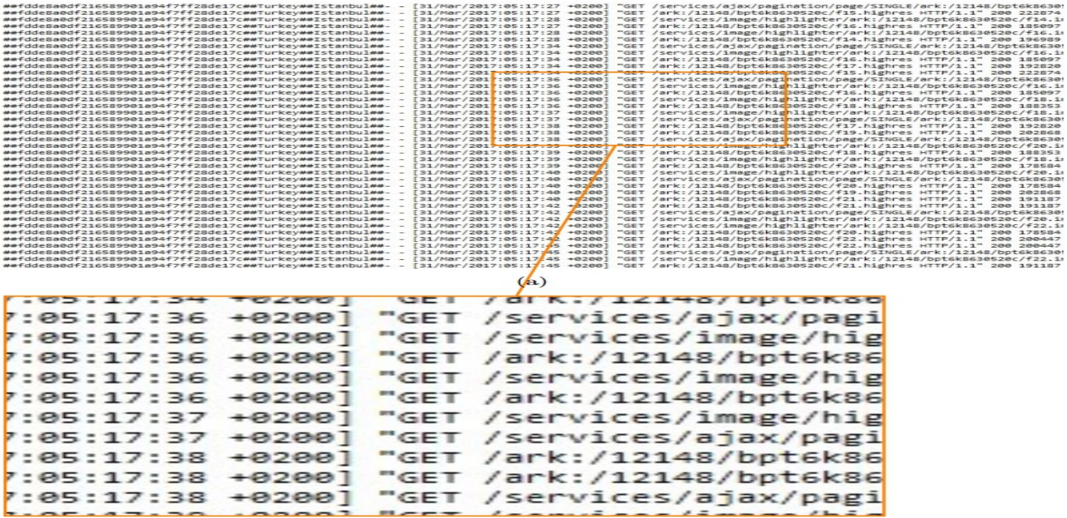
Exponential number of events in Digital Library logs

- Is clustering a solution for the huge number of logs?
- Can we find representative clusters (users types/tasks)?
- Which clustering method should be proposed for a large, complex and unstructured logs?

Handling Digital Library users' logs

How many logs are required to generate relevant models for both users and designers?

Logs quality



Logs prepocessing

1) Data visualization

- ELK services to visualise queries.
- Global view of users' queries.
- $\sim 500M$ every month.
- April 2017.

- **Web design queries** (Javascript, CSS...)
- **HTML queries** (static web pages $\rightarrow$ collections navigation...)
- **SRU queries** (Search and Retrieve via URL $\rightarrow$ search engine)
- **ARK queries** (Collections identification)

ELK services refers to `https://www.elastic.co/fr/elastic-stack/`

Logs prepocessing

1) Data visualization

- ELK services to visualise queries.
- Global view of users' queries.
- $\sim 500M$ every month.
- April 2017.

2) Outliers Detection

- Filtering queries from the bots-crawlers
- Deleting irrelevant queries: css, js
- Deleting $\sim$ 60% of the queries

Logs prepocessing

1) Data visualization

- ELK services to visualise queries.
- Global view of users' queries.
- $\sim 500M$ every month.
- April 2017.

2) Outliers Detection

- Filtering queries from the bots-crawlers
- Deleting irrelevant queries: css, js
- Deleting $\sim$ 60% of the queries

3) Users' Queries Tagging

- Standard convention to tag queries.
- Normalisation using standard activity's name.
- 9 activity names.

Logs prepocessing

1) Data visualization

- ELK services to visualise queries.
- Global view of users' queries.
- $\sim 500M$ every month.
- April 2017.

2) Outliers Detection

- Filtering queries from the bots-crawlers
- Deleting irrelevant queries: css, js
- Deleting $\sim$ 60% of the queries

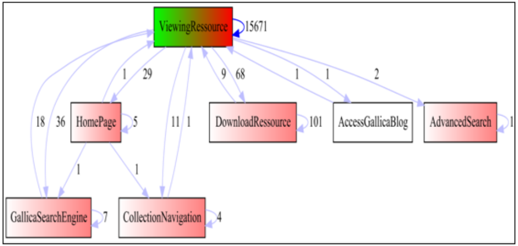
3) Users' Queries Tagging

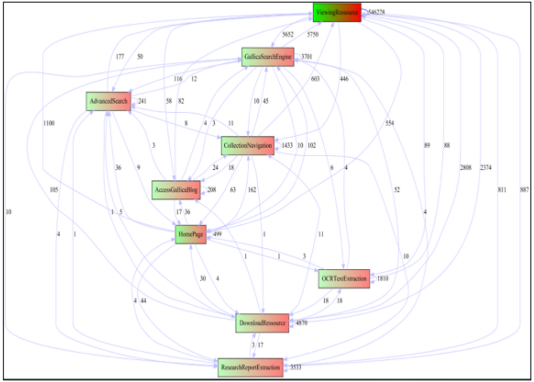
- Standard convention to tag queries.
- Normalisation using standard activity's name.
- 9 activity names.

4) Sessionization

- Dividing all the users' queries into sessions.
- Session: a 1-hour navigation of the same user (IP address).

Findings (2 clusters for the first 20,000 traces)





Outline

Conclusion

- Trace clustering techniques improve both quality of process models, as well as reduces the amount of time needed to discover a single model.
- Data clustering algorithms group data points based on their distance in a feature vector space. However, they are unable to perform strongly under process-oriented event logs.
- Researchers attempted to overcome this deficiency by adopting data clustering ideas in the process mining context.


D'ici, on voit + loin !
La Rochelle Université
univ-larochelle.fr