Mining Usage Patterns from a Repository of Scientific Workflows

Claudia Diamantini, Domenico Potena, Emanuele Storti

DII, Università Politecnica delle Marche, Ancona, Italy

SAC 2012, Riva del Garda, March 26-30
Introduction

**Process mining** techniques allow for extracting knowledge from event logs, i.e. traces of running processes:

- audit trails of a workflow management system
- transaction logs of an enterprise resource planning system

Main applications:

- **process discovery**: what is really happening?
- conformance check: are we doing what was agreed upon?
- process extension: how can we redesign the process?
Introduction

**Process mining** techniques allow for extracting knowledge from event logs, i.e. traces of running processes:
- audit trails of a workflow management system
- transaction logs of an enterprise resource planning system

Main applications:
- process discovery: what is really happening?
- **conformance check**: are we doing what was agreed upon?
- process extension: how can we redesign the process?
Introduction

**Process mining** techniques allow for extracting knowledge from event logs, i.e. traces of running processes:

- audit trails of a workflow management system
- transaction logs of an enterprise resource planning system

Main applications:

- process discovery: what is really happening?
- conformance check: are we doing what was agreed upon?
- **process extension**: how can we redesign the process?
Motivation

Little work about how to extract knowledge from a set of models (i.e.: process schemas)

- understanding of common patterns of usage (best/worst practices)
- support in process design
- process integration (from multiple sources)
- process optimization/re-engineering
- indexing of processes and their retrieval
Methodology

Processes schemas have an inherent graph structure: many graph-mining tools available

Approach

1. representation of original processes into graphs
2. usage of graph-mining techniques to extract SUBs

Which representation form?
Which specific technique?
Graph clustering techniques (sub-processes are clusters) with a hierarchical approach (various levels of abstractions, more/less specificity)
Methodology

Processes schemas have an inherent graph structure: many graph-mining tools available

Approach

1. representation of original processes into graphs
2. usage of graph-mining techniques to extract SUBs

Which representation form?

Which specific technique?

Graph clustering techniques (sub-processes are clusters) with a hierarchical approach (various levels of abstractions, more/less specificity)
Methodology

Processes schemas have an inherent graph structure: many graph-mining tools available

Approach

1. representation of original processes into graphs
2. usage of graph-mining techniques to extract SUBs

Which representation form?

Which specific technique?

Graph clustering techniques (sub-processes are clusters) with a hierarchical approach (various levels of abstractions, more/less specificity)
Methodology

Processes schemas have an inherent graph structure: many graph-mining tools available

**Approach**

1. representation of original processes into graphs
2. usage of graph-mining techniques to extract SUBs

Which representation form?

Which specific technique?

**Graph clustering** techniques (sub-processes are clusters) with a **hierarchical** approach (various levels of abstractions, more/less specificity)
**SUBDUE [Joyner, 2000]**

- graph-based hierarchical clustering algorithm
- suited for discrete-valued and structured data
- searches for substructures (i.e., subgraphs) that best compress the input graph, according to MDL

**Comments:**

- to compress G with a SUB: to replace every occurrence of SUB in G with a single new node
- SUB best compresses G: num. of bits needed to represent G after the compression is minimum
SUBDUE

SUBDUE [Joyner, 2000]

- graph-based hierarchical clustering algorithm
- suited for discrete-valued and structured data
- searches for substructures (i.e., subgraphs) that best compress the input graph, according to MDL

Comments:

- **to compress G with a SUB**: to replace every occurrence of SUB in G with a single new node
- **SUB best compresses G**: num. of bits needed to represent G after the compression is minimum
SUBDUE [Joyner, 2000]

- graph-based hierarchical clustering algorithm
- suited for discrete-valued and structured data
- searches for substructures (i.e., subgraphs) that best compress the input graph, according to MDL

Comments:

- to compress $G$ with a SUB: to replace every occurrence of SUB in $G$ with a single new node
- **SUB best compresses $G$**: num. of bits needed to represent $G$ after the compression is minimum
1. **searches for the best SUB** (i.e., that minimizes the Description Length of the graph)

2. compresses the graph by using the best SUB

Subsequent SUBs may be defined in terms of previously defined SUBs. Iterations of this basic step results in a **lattice**
1. searches for the best SUB (i.e., that minimizes the Description Length of the graph)
2. compresses the graph by using the best SUB

Subsequent SUBs may be defined in terms of previously defined SUBs. Iterations of this basic step results in a lattice.
1. searches for the best SUB (i.e., that minimizes the Description Length of the graph)
2. compresses the graph by using the best SUB

Subsequent SUBs may be defined in terms of previously defined SUBs. Iterations of this basic step results in a lattice.
1. searches for the best **SUB** (i.e., that minimizes the Description Length of the graph)
2. compresses the graph by using the best **SUB**

Subsequent SUBs may be defined in terms of previously defined SUBs. Iterations of this basic step results in a **lattice**
1. searches for the best SUB (i.e., that minimizes the Description Length of the graph)

2. compresses the graph by using the best SUB

Subsequent SUBs may be defined in terms of previously defined SUBs. Iterations of this basic step results in a lattice.
1. **searches for the best SUB** (i.e., that minimizes the Description Length of the graph)

2. **compresses the graph** by using the best SUB

Subsequent SUBs may be defined in terms of previously defined SUBs. Iterations of this basic step results in a **lattice**
1. search for the best SUB (i.e., that minimizes the Description Length of the graph)
2. compresses the graph by using the best SUB

Subsequent SUBs may be defined in terms of previously defined SUBs. Iterations of this basic step results in a lattice.
Traditional cluster **evaluation** is not applicable to hierarchic domains

\[
\text{quality} = \frac{\text{interClusterDistance}}{\text{intraClusterDistance}}
\]

Desirable features of a good clustering:
- few and big clusters (large coverage, good generality)
- minimal/no overlap (better defined concepts)

**New indexes**
- **completeness**: % of original nodes/edges still present in the final lattice
- **representativeness** (of a SUB): % of input graphs holding the SUB at least once
- **significance**: qualitative measure of the meaningfulness of a cluster w.r.t. the domain and application
Traditional cluster **evaluation** is not applicable to hierarchic domains

\[
\text{quality} = \frac{\text{interClusterDistance}}{\text{intraClusterDistance}}
\]

Desirable features of a good clustering:
- few and big clusters (large coverage, good generality)
- minimal/no overlap (better defined concepts)

**New indexes**

- **completeness**: % of original nodes/edges still present in the final lattice
- **representativeness** (of a SUB): % of input graphs holding the SUB at least once
- **significance**: qualitative measure of the meaningfulness of a cluster w.r.t. the domain and application
Traditional cluster **evaluation** is not applicable to hierarchic domains

$$\text{quality} = \frac{\text{interClusterDistance}}{\text{intraClusterDistance}}$$

Desirable features of a good clustering:
- few and big clusters (large coverage, good generality)
- minimal/no overlap (better defined concepts)

**New indexes**

- **completeness**: % of original nodes/edges still present in the final lattice
- **representativeness** (of a SUB): % of input graphs holding the SUB at least once
- **significance**: qualitative measure of the meaningfulness of a cluster w.r.t. the domain and application
Traditional cluster evaluation is not applicable to hierarchic domains

\[
\text{quality} = \frac{\text{interClusterDistance}}{\text{intraClusterDistance}}
\]

Desirable features of a good clustering:

- few and big clusters (large coverage, good generality)
- minimal/no overlap (better defined concepts)

New indexes

- **completeness**: % of original nodes/edges still present in the final lattice
- **representativeness** (of a SUB): % of input graphs holding the SUB at least once
- **significance**: qualitative measure of the meaningfulness of a cluster w.r.t. the domain and application
Representation

Approach

1. representation of original processes into graphs
2. usage of graph-mining techniques to extract SUBs

Common operators:
- sequence
- split-AND (parallel split), split-XOR (exclusive choice)
- join-AND (synchronization), join-XOR (simple merge)

How to map the process to the graph? Several choices
- lowest level of compactness
- every operator is represented by a node called *operator*, linked to another node specifying its *type* (AND/OR/SEQ) and to its *operands*
- join/split are distinguishable by the number of in/out-going arcs
- medium level of compactness
- the closest to the original representation
- join/split are replaced by different nodes, one for each kind of operator
Representation models: C rule

- highest level of compactness
- implicit representation of operators
- multiple alternative graphs in case of SPLIT-XOR or JOIN-XOR
- arcs are labeled to preserve information about domain/range nodes
the three representations hold the same information, with no loss

easy extension with more attributes (actor name/role, info about
time/resources, ...):
- attribute name → edge
- value → a node

Need of experimentations to assess which one performs best w.r.t.
the quality of clustering.
Evaluation

Comments

- the three representations hold the same information, with no loss
- easy extension with more attributes (actor name/role, info about time/resources, ...):
  - attribute name → edge
  - value → a node

Need of experimentations to assess which one performs best w.r.t. the quality of clustering.
the three representations hold the same information, with no loss

easy extension with more attributes (actor name/role, info about
time/resources, ...):

- attribute name $\rightarrow$ edge
- value $\rightarrow$ a node

Need of **experimentations** to assess which one performs best w.r.t.
the quality of clustering.
**Experiment**

**Dataset:**
- 258 processes (XScufl language for Taverna framework)
- e-Science WF for distributed computation over scientific data
- applications: local tools, Web Services, scripts

**Setting:**
- WF extraction/parsing
- preprocessing
- translation into graphs (A,B,C rules)
- SUBDUE execution
Experimental results

Output lattice (fragment)

Red boxes = top level SUBs
### Experimental results

<table>
<thead>
<tr>
<th></th>
<th>A rule</th>
<th>B rule</th>
<th>C rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size (nodes)</td>
<td>5618</td>
<td>2871</td>
<td>2414</td>
</tr>
<tr>
<td>SUBs (num)</td>
<td>671</td>
<td>611</td>
<td>580</td>
</tr>
<tr>
<td>top SUBs</td>
<td>2.24%</td>
<td>52.37%</td>
<td>52.93%</td>
</tr>
<tr>
<td>Completeness</td>
<td>95.41%</td>
<td>90.90%</td>
<td>90.27%</td>
</tr>
<tr>
<td>Representativeness</td>
<td>99.22%</td>
<td>31.78%</td>
<td>23.26%</td>
</tr>
<tr>
<td>Significance of top level clusters</td>
<td>–</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>Execution time (hh:mm)</td>
<td>03:38</td>
<td>00:06</td>
<td>00:03</td>
</tr>
</tbody>
</table>

### Considerations:
- A model: low significance, high execution time
- No definitively best choice between B and C:
  - B model: higher representativeness → indexing
  - C model: higher significance → usage pattern discovery
Experimental results

<table>
<thead>
<tr>
<th></th>
<th>A rule</th>
<th>B rule</th>
<th>C rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size (nodes)</td>
<td>5618</td>
<td>2871</td>
<td>2414</td>
</tr>
<tr>
<td>SUBs (num)</td>
<td>671</td>
<td>611</td>
<td>580</td>
</tr>
<tr>
<td>top SUBs</td>
<td>2.24%</td>
<td>52.37%</td>
<td>52.93%</td>
</tr>
<tr>
<td>Completeness</td>
<td>95.41%</td>
<td>90.90%</td>
<td>90.27%</td>
</tr>
<tr>
<td>Representativeness</td>
<td>99.22%</td>
<td>31.78%</td>
<td>23.26%</td>
</tr>
<tr>
<td>Significance of top level clusters</td>
<td>–</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>Execution time (hh:mm)</td>
<td>03:38</td>
<td>00:06</td>
<td>00:03</td>
</tr>
</tbody>
</table>

Considerations:

- A model: low significance, high execution time
- No definitively best choice between B and C:
  - B model: higher representativeness → indexing
  - C model: higher significance → usage pattern discovery
### Experimental results

<table>
<thead>
<tr>
<th></th>
<th>A rule</th>
<th>B rule</th>
<th>C rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size (nodes)</td>
<td>5618</td>
<td>2871</td>
<td>2414</td>
</tr>
<tr>
<td>SUBs (num)</td>
<td>671</td>
<td>611</td>
<td>580</td>
</tr>
<tr>
<td>top SUBs</td>
<td>2.24%</td>
<td>52.37%</td>
<td>52.93%</td>
</tr>
<tr>
<td>Completeness</td>
<td>95.41%</td>
<td>90.90%</td>
<td>90.27%</td>
</tr>
<tr>
<td>Representativeness (first 10 top SUBs)</td>
<td>99.22%</td>
<td>31.78%</td>
<td>23.26%</td>
</tr>
<tr>
<td>Significance of top level clusters</td>
<td>–</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>Execution time (hh:mm)</td>
<td>03:38</td>
<td>00:06</td>
<td>00:03</td>
</tr>
</tbody>
</table>

**Considerations:**

- **A model:** low significance, high execution time
- No definitively best choice between B and C:
  - **B model:** higher representativeness → indexing
  - **C model:** higher significance → usage pattern discovery
## Experimental results

<table>
<thead>
<tr>
<th></th>
<th>A rule</th>
<th>B rule</th>
<th>C rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size (nodes)</td>
<td>5618</td>
<td>2871</td>
<td>2414</td>
</tr>
<tr>
<td>SUBs (num)</td>
<td>671</td>
<td>611</td>
<td>580</td>
</tr>
<tr>
<td>top SUBs</td>
<td>2.24%</td>
<td>52.37%</td>
<td>52.93%</td>
</tr>
<tr>
<td>Completeness</td>
<td>95.41%</td>
<td>90.90%</td>
<td>90.27%</td>
</tr>
<tr>
<td>Representativeness</td>
<td>99.22%</td>
<td>31.78%</td>
<td>23.26%</td>
</tr>
<tr>
<td>Significance of top</td>
<td>−</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>level clusters</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Execution time</td>
<td>03:38</td>
<td>00:06</td>
<td>00:03</td>
</tr>
</tbody>
</table>

### Considerations:

- **A model:** low significance, high execution time
- No definitively best choice between B and C:
  - **B model:** higher representativeness → indexing
  - **C model:** higher significance → usage pattern discovery
Experimental results

<table>
<thead>
<tr>
<th></th>
<th>A rule</th>
<th>B rule</th>
<th>C rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size (nodes)</td>
<td>5618</td>
<td>2871</td>
<td>2414</td>
</tr>
<tr>
<td>SUBs (num)</td>
<td>671</td>
<td>611</td>
<td>580</td>
</tr>
<tr>
<td>top SUBs</td>
<td>2.24%</td>
<td>52.37%</td>
<td>52.93%</td>
</tr>
<tr>
<td>Completeness</td>
<td>95.41%</td>
<td>90.90%</td>
<td>90.27%</td>
</tr>
<tr>
<td>Representativeness</td>
<td>99.22%</td>
<td>31.78%</td>
<td>23.26%</td>
</tr>
<tr>
<td>Significance of top</td>
<td>–</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>level clusters</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Execution time (hh:mm)</td>
<td>03:38</td>
<td>00:06</td>
<td>00:03</td>
</tr>
</tbody>
</table>

Considerations:

- A model: low significance, high execution time
- No definitively best choice between B and C:
  - B model: higher representativeness → indexing
  - C model: higher significance → usage pattern discovery
Experimental results

<table>
<thead>
<tr>
<th></th>
<th>A rule</th>
<th>B rule</th>
<th>C rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size (nodes)</td>
<td>5618</td>
<td>2871</td>
<td>2414</td>
</tr>
<tr>
<td>SUBs (num)</td>
<td>671</td>
<td>611</td>
<td>580</td>
</tr>
<tr>
<td>top SUBs</td>
<td>2.24%</td>
<td>52.37%</td>
<td>52.93%</td>
</tr>
<tr>
<td>Completeness</td>
<td>95.41%</td>
<td>90.90%</td>
<td>90.27%</td>
</tr>
<tr>
<td>Representativeness (first 10 top SUBs)</td>
<td>99.22%</td>
<td>31.78%</td>
<td>23.26%</td>
</tr>
<tr>
<td>Significance of top level clusters</td>
<td>–</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>Execution time (hh:mm)</td>
<td>03:38</td>
<td>00:06</td>
<td>00:03</td>
</tr>
</tbody>
</table>

Considerations:

- A model: low significance, high execution time
- No definitively best choice between B and C:
  - B model: higher representativeness → indexing
  - C model: higher significance → usage pattern discovery
## Experimental results

<table>
<thead>
<tr>
<th></th>
<th>A rule</th>
<th>B rule</th>
<th>C rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size (nodes)</td>
<td>5618</td>
<td>2871</td>
<td>2414</td>
</tr>
<tr>
<td>SUBs (num)</td>
<td>671</td>
<td>611</td>
<td>580</td>
</tr>
<tr>
<td>top SUBs</td>
<td>2.24%</td>
<td>52.37%</td>
<td>52.93%</td>
</tr>
<tr>
<td>Completeness</td>
<td>95.41%</td>
<td>90.90%</td>
<td>90.27%</td>
</tr>
<tr>
<td>Representativeness (first 10 top SUBs)</td>
<td>99.22%</td>
<td>31.78%</td>
<td>23.26%</td>
</tr>
<tr>
<td>Significance of top level clusters</td>
<td>–</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>Execution time (hh:mm)</td>
<td>03:38</td>
<td>00:06</td>
<td>00:03</td>
</tr>
</tbody>
</table>

### Considerations:

- A model: low significance, high execution time
- No definitively best choice between B and C:
  - B model: higher representativeness → indexing
  - C model: higher significance → usage pattern discovery
## Experimental results

<table>
<thead>
<tr>
<th></th>
<th>A rule</th>
<th>B rule</th>
<th>C rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size (nodes)</td>
<td>5618</td>
<td>2871</td>
<td>2414</td>
</tr>
<tr>
<td>SUBs (num)</td>
<td>671</td>
<td>611</td>
<td>580</td>
</tr>
<tr>
<td>top SUBs</td>
<td>2.24%</td>
<td>52.37%</td>
<td>52.93%</td>
</tr>
<tr>
<td>Completeness</td>
<td>95.41%</td>
<td>90.90%</td>
<td>90.27%</td>
</tr>
<tr>
<td>Representativeness (first 10 top SUBs)</td>
<td>99.22%</td>
<td>31.78%</td>
<td>23.26%</td>
</tr>
<tr>
<td>Significance of top level clusters</td>
<td>–</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>Execution time (hh:mm)</td>
<td>03:38</td>
<td>00:06</td>
<td>00:03</td>
</tr>
</tbody>
</table>

### Considerations:

- A model: low significance, high execution time
- No definitively best choice between B and C:
  - B model: higher representativeness $\rightarrow$ indexing
  - C model: higher significance $\rightarrow$ usage pattern discovery
Use Case

Scenario: a scientist wants to use web service $X$ for DNA alignment

Baseline: browse $my$ Experiment to find every workflow with $X$:
- too many results
- given a workflow: common practice or exception?

New approach: search for $X$ in the lattice’s SUBs:
- find all usage patterns for $X$ (i.e., top SUBs containing $X$)
- analyse their representativeness
- choose one of them or browse more in depth the lattice
Use Case

Scenario: a scientist wants to use web service $X$ for DNA alignment

**Baseline:** browse *my Experiment* to find every workflow with $X$:
- too many results
- given a workflow: common practice or exception?

**New approach:** search for $X$ in the lattice’s SUBs:
- find all usage patterns for $X$ (i.e., top SUBs containing $X$)
- analyse their representativeness
- choose one of them or browse more in depth the lattice
Use Case

Scenario: a scientist wants to use web service $X$ for DNA alignment

Baseline: browse $my$ Experiment to find every workflow with $X$:
- too many results
- given a workflow: common practice or exception?

**New approach: search for $X$ in the lattice’s SUBs:**
1. find all usage patterns for $X$ (i.e., top SUBs containing $X$)
2. analyse their representativeness
3. choose one of them or browse more in depth the lattice
Use Case

Scenario: a scientist wants to use web service $X$ for DNA alignment

Baseline: browse $my$ Experiment to find every workflow with $X$:
  - too many results
  - given a workflow: common practice or exception?

New approach: search for $X$ in the lattice’s SUBs:
1. find all usage patterns for $X$ (i.e., top SUBs containing $X$)
2. analyse their representativeness
3. choose one of them or browse more in depth the lattice
Use Case

Scenario: a scientist wants to use web service $X$ for DNA alignment

Baseline: browse *my* Experiment to find every workflow with $X$:
- too many results
- given a workflow: common practice or exception?

**New approach: search for $X$ in the lattice’s SUBs:**
1. find all usage patterns for $X$ (i.e., top SUBs containing $X$)
2. analyse their representativeness
3. choose one of them or browse more in depth the lattice
Use Case

Scenario: a scientist wants to use web service X for DNA alignment

Baseline: browse myExperiment to find every workflow with X:
  - too many results
  - given a workflow: common practice or exception?

New approach: search for X in the lattice’s SUBs:
  1. find all usage patterns for X (i.e., top SUBs containing X)
  2. analyse their representativeness
  3. choose one of them or browse more in depth the lattice
Applications:

- **reference during process design** (learn-by-example)
- process retrieval: find every *my* Experiment WF containing such a pattern
Use case (cont’d)

Applications:
- reference during process design (learn-by-example)
- **process retrieval**: find every *my* Experiment WF containing such a pattern
A graph-based clustering technique (SUBDUE) to recognize the most frequent/common subprocesses among a set of workflows/process schemas.

Comments

- several representation alternatives for application of SUBDUE
- evaluations on specialized e-Science domain
- useful to find typical patterns and schemas of usage for tools/tasks
Conclusion

Applications:

- recognition of **reference processes** (common/best/worst practices)
- organize a **process repository** (by indexing processes through substructures)
- **enterprise integration**: find similarities (differences, overlapping, complementariness) in BP in different companies

**Future work:**

- extending experimentations, especially with (real) Business Processes
- management of heterogeneity (syntactic/semantic, level of granularity)
Mining Usage Patterns from a Repository of Scientific Workflows

Claudia Diamantini, Domenico Potena, Emanuele Storti

DII, Universitá Politecnica delle Marche, Ancona, Italy

SAC 2012, Riva del Garda, March 26-30
Subdue uses a variant of beam search (heuristic)
Best SUB minimizes the Description Length of the graph

1. **Search the best SUB**
   - **Init**: set of SUBs consisting of all uniquely labeled vertices
   - extend each SUB in all possible ways by a single edge and a vertex
   - order SUBs according to MDL principle
   - keep only the top $n$ SUBs
   - **End**: exhaustion of search space or user constraints
Subdue uses a variant of beam search (heuristic)
Best SUB minimizes the Description Length of the graph

1. Search the best SUB
   - *Init*: set of SUBs consisting of all uniquely labeled vertices
   - extend each SUB in all possible ways by a single edge and a vertex
   - order SUBs according to MDL principle
   - keep only the top $n$ SUBs
   - *End*: exhaustion of search space or user constraints
Subdue uses a variant of beam search (heuristic)
Best SUB minimizes the Description Length of the graph

1 Search the best SUB
- *Init*: set of SUBs consisting of all uniquely labeled vertices
- extend each SUB in all possible ways by a single edge and a vertex
- order SUBs according to MDL principle
- keep only the top $n$ SUBs
- *End*: exhaustion of search space or user constraints
Subdue uses a variant of beam search (heuristic)
Best SUB minimizes the Description Length of the graph

1. **Search the best SUB**
   - *Init*: set of SUBs consisting of all uniquely labeled vertices
   - extend each SUB in all possible ways by a single edge and a vertex
   - order SUBs according to MDL principle
   - keep only the top $n$ SUBs
   - *End*: exhaustion of search space or user constraints
Subdue uses a variant of beam search (heuristic)
Best SUB minimizes the Description Length of the graph

1. **Search the best** SUB
   - *Init*: set of SUBs consisting of all uniquely labeled vertices
   - extend each SUB in all possible ways by a single edge and a vertex
   - order SUBs according to MDL principle
   - keep only the top $n$ SUBs
   - *End*: exhaustion of search space or user constraints
Subdue uses a variant of beam search (heuristic)
Best SUB minimizes the Description Length of the graph

1. **Search the best SUB**
   - *Init*: set of SUBs consisting of all uniquely labeled vertices
   - extend each SUB in all possible ways by a single edge and a vertex
   - order SUBs according to MDL principle
   - keep only the top $n$ SUBs
   - *End*: exhaustion of search space or user constraints
Subdue uses a variant of beam search (heuristic)
Best SUB minimizes the Description Length of the graph

Search the best SUB

- **Init**: set of SUBs consisting of all uniquely labeled vertices
- extend each SUB in all possible ways by a single edge and a vertex
- order SUBs according to MDL principle
- keep only the top \(n\) SUBs
- **End**: exhaustion of search space or user constraints
Subdue uses a variant of beam search (heuristic)
Best SUB minimizes the Description Length of the graph

**1. Search the best SUB**
- \( \text{Init: set of SUBs consisting of all uniquely labeled vertices} \)
- extend each SUB in all possible ways by a single edge and a vertex
- order SUBs according to MDL principle
- keep only the top \( n \) SUBs
- \( \text{End: exhaustion of search space or user constraints} \)

**2. Compress the graph with the best SUB**
Subdue uses a variant of beam search (heuristic)
Best SUB minimizes the Description Length of the graph

1. **Search the best SUB**
   - *Init*: set of SUBs consisting of all uniquely labeled vertices
   - extend each SUB in all possible ways by a single edge and a vertex
   - order SUBs according to MDL principle
   - keep only the top $n$ SUBs
   - *End*: exhaustion of search space or user constraints

2. **Compress the graph with the best SUB**
Subdue uses a variant of beam search (heuristic)
Best SUB minimizes the Description Length of the graph

1. Search the best SUB
   - *Init*: set of SUBs consisting of all uniquely labeled vertices
   - extend each SUB in all possible ways by a single edge and a vertex
   - order SUBs according to MDL principle
   - keep only the top $n$ SUBs
   - *End*: exhaustion of search space or user constraints

2. Compress the graph with the best SUB

Subsequent SUBs may be defined in terms of previously defined SUBs

Iterations of this basic step result in a **lattice**