National Research University Higher School of Economics
Perm, Russian Federation

HP-GRAPH: DEFINITION AND APPROACHES TO OPTIMIZING ALGORITHMS FOR GRAPHS

Nikolai M. Suvorov
Lyudmila N. Lyadova

ITA, 2020
RESEARCH RELEVANCE:

MULTI-MODELING APPROACH

MODEL BUILDING TOOLS
Model Building Tools

MODEL ANALYSIS TOOLS
Model Analysis Tools

BEHAVIOR MODELS
Behavior Models

PROCESSING MODELS
Processing Models

HARDWARE MODELS
Hardware Models

SYSTEM GENERATOR
System Generator

CONFIGURATION GENERATOR
Configuration Generator

Runtime System

DESIGN DEVELOPMENT

RUNTIME DEVELOPMENT
RESEARCH RELEVANCE:
RESEARCH AREA PROBLEM
RESEARCH RELEVANCE:
RESEARCH AREA PROBLEM
RESEARCH RELEVANCE: RESEARCH AREA PROBLEM

Queuing Systems Models

Mathematical Modeling

Petri Nets

Simulation Models

M₀ = 3p₁ + 3p₆ + 2p₇ + 2p₈
M₁ = 2p₁ + 2p₂ + 3p₆ + p₇ + 2p₈
M₂ = 2p₂ + p₃ + 2p₄ + p₆ + p₇ + p₈
M₃ = 3p₁ + 2p₄ + p₆ + 2p₇
M₄ = 3p₁ + p₅ + 2p₆ + 2p₇
M₅ = p₁ + p₃ + 3p₆ + 2p₇
M₆ = p₁ + 2p₂ + p₃ + 2p₄ + 2p₆ + p₇ + p₈
M₇ = 2p₂ + 2p₃ + 3p₆ + 2p₇
M₈ = p₁ + 2p₄ + p₆ + 2p₇
M₉ = 3p₁ + p₅ + 2p₆ + 2p₇
M₁₀ = 2p₁ + 2p₂ + 2p₄ + p₆ + p₇
M₁₁ = 3p₁ + p₄ + 5p₆ + p₇
M₁₂ = 2p₂ + 2p₃ + 3p₆
M₁₃ = p₁ + 2p₂ + 2p₄ + p₆
M₁₄ = 3p₁ + 2p₄ + p₅

good marking   dangerous marking   bad marking   dead marking
RESEARCH RELEVANCE:
RESEARCH AREA PROBLEM
RESEARCH RELEVANCE:
TRANSFORMATIONS
RESEARCH RELEVANCE: TRANSFORMATIONS
RESEARCH RELEVANCE:
TRANSFORMATIONS
RESEARCH RELEVANCE: LANGUAGE FOCUSED APPROACH
Graph model should:

- Allow us to **formalize the description of all elements**, included in visual languages.
- Allow us to **implement general modeling principles**.
- Provide a possibility to **implement transformations** of various types of models.
- **Optimize the algorithms** that should be implemented for modeling and solving tasks with models.
RESEARCH PURPOSE AND TASKS

Purpose:
Defining a *new graph formalism* that can be used as a basis for a DSM platform development, providing a possibility to perform multi-level and multi-aspect modeling.

Tasks:
- Analysis of different types of graphs to determine how well they meet the mentioned requirements.
- Development of a new graph formalism and evaluating and comparing it with existing ones.
- Development of algorithms for the new graph formalism.
- Development of a visual editor object model.
- Development of a program, demonstrating the practical significance of the selected graph formalism.
GRAPH MODEL DEFINITION:
SETS OF HP-GRAPH

HP-graph is an ordered triple $G = (P, V, W)$:

- $P = \{\pi_1, \ldots, \pi_n\}$ is a set of external poles of the graph.
- $V = \{v_1, \ldots, v_m\}$ is a non-empty set of mutually disjoint vertices, consisting of internal poles.
- $W = \{w_1, \ldots, w_l\}$ is a set of hyperedges, consisting of poles.
- $Pol$ is a set of all poles of the graph.
GRAPH MODEL DEFINITION:
SETS OF HP-GRAH

HP-graph is an ordered triple $G = (P, V, W)$:

- $P = \{\pi_1, \ldots, \pi_n\}$ is a set of external poles of the graph.
- $V = \{v_1, \ldots, v_m\}$ is a non-empty set of mutually disjoint vertices, consisting of internal poles.
- $W = \{w_1, \ldots, w_l\}$ is a set of hyperedges, consisting of poles.
- $Pol$ is a set of all poles of the graph.
HP-graph is an ordered triple $G = (P, V, W)$:

- $P = \{\pi_1, \ldots, \pi_n\}$ is a set of external poles of the graph.
- $V = \{v_1, \ldots, v_m\}$ is a non-empty set of mutually disjoint vertices, consisting of internal poles.
- $W = \{w_1, \ldots, w_l\}$ is a set of hyperedges, consisting of poles.
- $Pol$ is a set of all poles of the graph
GRAPH MODEL DEFINITION:
SETS OF HP-GRAPH

HP-graph is an ordered triple $G = (P, V, W)$:

- $P = \{\pi_1, \ldots, \pi_n\}$ is a set of external poles of the graph.
- $V = \{v_1, \ldots, v_m\}$ is a non-empty set of mutually disjoint vertices, consisting of internal poles.
- $W = \{w_1, \ldots, w_l\}$ is a set of hyperedges, consisting of poles.
- $\text{Pol}$ is a set of all poles of the graph.
GRAPH MODEL DEFINITION:
INPUT AND OUTPUT POLES

• All the *poles* can be input or output (or both)

• Each *vertex* of the graph \( v \in V \) is also represented by a set of *input* \( (I(v)) \) and *output* \( (O(v)) \) poles

• Each *edge* must contain at least one input pole and one output pole
## GRAPH MODEL DEFINITION:
### BASIC OPERATIONS OF GRAPH EDITOR

<table>
<thead>
<tr>
<th>Addition Operations</th>
<th>Removal Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ( v + p_1 ) – Addition of the inner pole to the vertex</td>
<td>1. ( v - p_1 ) – Removal of the inner pole from the node</td>
</tr>
<tr>
<td>2. ( G + v ) – Addition of the vertex to the graph</td>
<td>2. ( G - v ) – Removal of the node from the graph</td>
</tr>
<tr>
<td>3. ( G + w ) – Addition of the edge to the graph</td>
<td>3. ( G - w ) – Removal of the edge from the graph</td>
</tr>
<tr>
<td>4. ( w + p_1 ) – Addition of the inner pole to the edge</td>
<td>4. ( w - p_1 ) – Removal of the inner pole from the edge</td>
</tr>
<tr>
<td>5. ( w + p_2 ) – Addition of the outer pole to the edge</td>
<td>5. ( w - p_2 ) – Removal of the external pole from the edge</td>
</tr>
<tr>
<td>6. ( G + p_2 ) – Addition of the outer pole to the graph</td>
<td>6. ( G - p_2 ) – Removal of the outer pole from the graph</td>
</tr>
</tbody>
</table>
GRAPH MODEL DEFINITION:
DECOMPOSITION BY A NEW GRAPH

Vertex Decomposition by a new HP-graph

Edge Decomposition by a new HP-graph
GRAPH MODEL DEFINITION: DECOMPOSITION BY A NEW GRAPH

• Every edge can be opened by ordinary links between poles.

• For every edge $w \in W$, a set of links is defined: $E_w = \{e_1, \ldots, e_n\} \subset I(w) \times O(w)$, where every link is a pair $(p, r)$ provided that $p \in I(w)$, $r \in O(w)$.

• As $E_w \subset I(w) \times O(w)$, some input and output poles can be unconnected such as poles $p9[O]$ and $p11[I]$. 

Edge Decomposition by ordinary links
### COMPARING GRAPH MODELS

<table>
<thead>
<tr>
<th>Graph model</th>
<th>Representation in the HP-graph $G' = (P', V', W')$</th>
</tr>
</thead>
</table>
| Oriented Graph $G = (V, E)$ | $V = P' = V'$, where $\forall v' \in V'$: $|v'| = 1$  
$E = W'$, where $\forall w' \in W'$: $|w'| = 2$) |
| Hypergraph $G = (X, E)$   | $X = P' = V'$, where $\forall v' \in V'$: $|v'| = 1$  
$E = W'$ |
| Hi-graph $G = (X, E)$    | $\{x \mid x \in X \& |x| = 1\} = P' = V'$, where $\forall v' \in V'$: $|v'| = 1$  
$E \cup \{x \mid x \in X \& |x| > 1\} = W'$ |
| Metagraph $G = (V, MV, E)$ | $V = P' = V'$, where $\forall v' \in V'$: $|v'| = 1$  
$E \cup MV = W'$ |
| P-graph $G = (P, V, W)$   | $P = P'$  
$V = V'$  
$W = W'$, where $\forall w' \in W'$: $|w'| = 2$ |
• As is seen, the decomposition of edges and vertices is almost equal, therefore, it is possible to define a common opening algorithm for these structures.

• Let us define a set of structures
  $Str = V \cup W$.

• Hence, $str \in Str$ is a structure which can be either a vertex or an edge.

• For every structure $str$ several decoding operations can be defined:

  $Open_{str} \subset str \times G_{all}$

---

**Procedure DecomposeStructure**

```java
G = new HPGraph();

foreach p \in str. 
  if (p \in l(str)):
    I(G) = I(G) \cup p;
  if (p \in O(str)):
    O(G) = O(G) \cup p;
    Open_{str} = Open_{str} \cup (str, G)
```
Let us define a subgraph of an HP-graph:

A subgraph of the HP graph $G = (P, V, W)$ is an HP-graph $G' = (P', V', W')$ that is part of the graph $G$ and fulfills the condition $\text{Open'} \subseteq \text{Open}$.

A subgraph can contain vertices called incomplete (partials) whose sets of poles can only be part of the sets of poles of the vertices of the original graph.

Transformation can be divided into 2 parts:

- **Removal of a subgraph, isomorphic to a pattern**
- **Addition of a replacement graph to the original graph**

---

**Function DeleteGraph(HostG, G_L)**

```plaintext
G' = Find_Isomorphic_Subgraph(HostG, G_L);
partials = {};
foreach $w' \in W(G')$:
    $W(\text{HostG}) = W(\text{HostG}) \setminus \{w'\}$;
foreach $v' \in V(G')$:
    if $(v' \in V(\text{HostG}))$:
        $V(\text{HostG}) = V(\text{HostG}) \setminus \{v'\}$;
    else:
        partials = partials $\cup \{v'\}$;
foreach $p' \in P(G')$:
    if $(\exists w \in W(\text{HostG})[p' \in w])$:
        $P(\text{HostG}) = P(\text{HostG}) \setminus p'$;
return partials;
```
Let us define a subgraph of an HP-graph:

A subgraph of the HP graph $G = (P, V, W)$ is an HP-graph $G' = (P', V', W')$ that is part of the graph $G$ and fulfills the condition $\text{Open}' \subset \text{Open}$.

A subgraph can contain vertices called incomplete (partials) whose sets of poles can only be part of the sets of poles of the vertices of the original graph.

Transformation can be divided into 2 parts:

- Removal of a subgraph, isomorphic to a pattern
- Addition of a replacement graph to the original graph

Procedure AddGraph(HostG, $G_R$, partials)

```plaintext
foreach $p \in P(G_R)$:
    if $p \notin P(\text{HostG})$:
        $P(\text{HostG}) = P(\text{HostG}) \cup \{p\}$;

foreach $v \in V(G_R)$:
    if ($v \notin \text{Partials}$):
        $V(\text{HostG}) = V(\text{HostG}) \cup \{v\}$;

foreach $w \in W(G_R)$:
    $W(\text{HostG}) = W(\text{HostG}) \cup \{w\}$;
```
TIME COMPLEXITY:
LAYER STRUCTURE FOR OPTIMIZING ALGORITHMS

Graph Layer

Vertices and Edges Layer

Poles Layer
TIME COMPLEXITY:
LAYER STRUCTURE FOR OPTIMIZING ALGORITHMS

Graph Layer

Vertices and Edges Layer

Poles Layer
TIME COMPLEXITY:
LAYER STRUCTURE FOR OPTIMIZING ALGORITHMS

Graph Layer

Vertices and Edges Layer

Poles Layer
**TIME COMPLEXITY:**

**LAYER STRUCTURE FOR OPTIMIZING ALGORITHMS**

**Graph Layer**

- **Vertices and Edges Layer**

<table>
<thead>
<tr>
<th></th>
<th>$v_1$</th>
<th>$v_2$</th>
<th>...</th>
<th>$v_j$</th>
<th>...</th>
<th>$v_{n-1}$</th>
<th>$v_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>$v_1$</strong></td>
<td>$a_{11}$</td>
<td>$a_{12}$</td>
<td>...</td>
<td>$a_{1,j}$</td>
<td>...</td>
<td>$a_{1,n-1}$</td>
<td>$a_{1,n}$</td>
</tr>
<tr>
<td><strong>...</strong></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td><strong>$v_i$</strong></td>
<td>$a_{i,1}$</td>
<td>$a_{i,2}$</td>
<td>...</td>
<td>$a_{i,j}$</td>
<td>...</td>
<td>$a_{2,n-1}$</td>
<td>$a_{2,n}$</td>
</tr>
<tr>
<td><strong>...</strong></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td><strong>$v_n$</strong></td>
<td>$a_{n,1}$</td>
<td>$a_{n,2}$</td>
<td>...</td>
<td>$a_{n,j}$</td>
<td>...</td>
<td>$a_{n,n-1}$</td>
<td>$a_{n,n}$</td>
</tr>
</tbody>
</table>
TIME COMPLEXITY: LAYER STRUCTURE FOR OPTIMIZING ALGORITHMS

Graph Layer

Vertices and Edges Layer

<table>
<thead>
<tr>
<th>A</th>
<th>v₁</th>
<th>v₂</th>
<th>...</th>
<th>vₖ</th>
<th>...</th>
<th>vₙ₋₁</th>
<th>vₙ</th>
</tr>
</thead>
<tbody>
<tr>
<td>v₁</td>
<td>a₁₁</td>
<td>a₁₂</td>
<td>...</td>
<td>a₁ₖ</td>
<td>...</td>
<td>a₁ₙ₋₁</td>
<td>a₁ₙ</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>vᵢ</td>
<td>aᵢ₁</td>
<td>aᵢ₂</td>
<td>...</td>
<td>aᵢₖ</td>
<td>...</td>
<td>aᵢₙ₋₁</td>
<td>aᵢₙ</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vₙ</td>
<td>aₙ₁</td>
<td>aₙ₂</td>
<td>...</td>
<td>aₙₖ</td>
<td>...</td>
<td>aₙₙ₋₁</td>
<td>aₙₙ</td>
</tr>
</tbody>
</table>

\[ G = \text{HostG}(); \]
\[ n = |V(G)|; \]
\[ A = \text{New Matrix}([n, n]); \]
\[ \text{for } i = 1 \text{ to } n \text{ step 1:} \]
\[ \text{for } j = 1 \text{ to } n \text{ step 1:} \]
\[ \text{for each } w \in W(G): \]
\[ a_{i,j} = P(v_i) \cap P(w) \neq \emptyset \text{ and } P(v_j) \cap P(w) \neq \emptyset \]

return \( A; \)
TIME COMPLEXITY:
LAYER STRUCTURE FOR OPTIMIZING ALGORITHMS

<table>
<thead>
<tr>
<th></th>
<th>( v_1 )</th>
<th>( v_2 )</th>
<th>...</th>
<th>( v_j )</th>
<th>...</th>
<th>( v_{n-1} )</th>
<th>( v_n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A )</td>
<td>( a_{12} )</td>
<td>...</td>
<td>( a_{1,j} )</td>
<td>...</td>
<td>( a_{1,n-1} )</td>
<td>( a_{1,n} )</td>
<td></td>
</tr>
</tbody>
</table>

\[
G = \text{HostG}() ;
\]
\[
n = |V(G)| ;
\]
\[
A = \text{New Matrix}([n, n]) ;
\]
\[
\text{for } i = 1 \text{ to } n \text{ step 1 :}
\]
\[
\text{for } j = 1 \text{ to } n \text{ step 1 :}
\]
\[
\text{for each } w \in W(G) :
\]
\[
a_{i,j} = \mathcal{O}(v_i) \cap P(w) \neq \emptyset \text{ and } \mathcal{L}(v_j) \cap P(w) \neq \emptyset
\]
\[
\text{return } A ;
\]
CONCLUSION

• The definition of the mathematical apparatus underlying the visual model editor was given above.

• Algorithms for decoding vertices and edges, as well as algorithms for performing transformations, were described.

• The HP-graph unites expressive possibilities of various types of graphs, therefore, algorithms that are designed for these types of graphs can also be implemented for HP graphs.

• The time complexity of model transformation algorithms can be reduced.

The paper proves that HP-graph allows the creation of a flexible visual model editor based on this graph formalism for a DSM-platform. Representing both vertices and links as sets of poles simplifies the object model of DSM editor and visual model editing algorithms.
THANKS FOR ATTENTION!

Suvorov Nikolai Mikhailovich
E-mail: SuvorovNM@gmail.com

Lyadova Lyudmila Nickolaevna
E-mail: LNLyadova@gmail.com
APPENDIX...
RESEARCH RELEVANCE:
LANGUAGE FOCUSED APPROACH
**RESEARCH RELEVANCE:**

**TRANSFORMATIONS**

**Use-Case Diagram**

**Activity Diagram**

**Sequence Diagram**
RESEARCH RELEVANCE:
TRANSFORMATIONS

**Activity Diagram**
- **Reader**:
  - Taking a book
  - Book registration

- **Librarian**:
  - Returning a book

**Use-Case Diagram**
- **Reader**
  - Taking a book
- **Librarian**
  - Returning a book

**Sequence Diagram**
- **Librarian**
  - EnterReaderData(reader_ticket)
  - Information about the reader
  - SearchBookInDb(Name, Author)
  - Book information
  - [Book in stock] MakeBookGivingAct(bookId, reader_ticket, return_date)
  - [Book not in stock] Cancel()

**Information System**
- **Librarian**
  - Enter Data about Reader in Program
  - Select the Desired Reader
  - Enter Information about Book

- **Reader**
  - Give Personal Information
  - Give Information about Desired Book

- **System**
  - Output all Readers by the Specified Parameters
  - [Reader Found]
    - [YES]
    - [NO]
  - Output Information about the Book and its Availability
  - [Book in Stock]
    - [ДА] [НЕТ]
  - Find Book in Book Depository
  - Enter Data about Book Giving
  - Create Book Giving Act
RESEARCH RELEVANCE: TRANSFORMATIONS

**Use-Case Diagram**

- **Reader**
  - Taking a book
  - Book registration
  - Returning a book

- **Librarian**
  - Book registration
  - Returning a book

**Activity Diagram**

- **Reader**
  - Give Personal Information
  - Give Information about Desired Book
  - Enter Information about Book
  - Output Information about the Book and its Availability

- **Librarian**
  - Enter Data about Reader in Program
  - Select the Desired Reader
  - Enter Information about Book
  - Output all Readers by the Specified Parameters

- **Information System**
  - [Reader Found]
  - [YES] [NO]
  - [Book in Stock]
  - [ДА] [НЕТ]
  - Find Book in Book Depository
  - Enter Data about Book Giving
  - Create Book Giving Act

**Sequence Diagram**

- **Librarian**
  - EnterReaderData(reader_ticket)
  - Information about the reader
  - SearchBookInDb(Name, Author)
  - Book information
  - [Book in stock] MakeBookGivingAct(bookId, reader_ticket, return_date)
  - [Book not in stock] Cancel()
EXPENDING TRANSFORMATIONS

G0

Center

G3

Processor

Center

RAM

Peripherals
REDUCING TRANSFORMATIONS

GL4

Vertex1: Processor
Center
Vertex3: Peripherals

Vertex2: RAM

GR4

Vertex1: Processor

Vertex2: RAM

Vertex3: Peripherals

Processor
RAM
Peripherals