Less is More: How to Tame a Very Large ER Diagram

Yannis Tzitzikas¹ and Jean-Luc Hainaut²

¹University of Crete (Adjunct Assistant Professor) & FORTH-ICS
²University of Namur (F.U.N.D.P.), Belgium

Work to be presented at ER’2005, Klagenfurt, Austria, Oct. 2005
Outline of the presentation

Motivation
   – Conceptual Diagrams: Why we use them? The problem of scale.

(A) Deriving top-k ER Diagrams (using Link Analysis)
   – EntityRank, B(iased)EntityRank

(B) Drawing top-k ER Diagrams

Conclusions
Conceptual Diagrams

• They are useful and widely used in
  – Requirements Engineering
  – Reverse Engineering
  – Knowledge Representation

• Examples
  – ER diagrams
  – UML class diagrams
  – SWeb ontologies
  – ...

But why we use them?

They aid communication

• customers vs analysts
• analysts vs domain experts
• analysts vs designers
• designers vs designers
• designers vs developers
• designers vs end users
• ...

But why we use them?

They aid communication
- customers vs analysts
- analysts vs domain experts
- analysts vs designers
- designers vs designers
- designers vs developers
- designers vs end users
- ...

A picture worths thousands of words
However, their usefulness **degrades rapidly** as they grow in size.

Very small excerpt of a schema (DICO1) with **456** entity and **812** relationship types.

Big schemas become more and more frequent:
- SAP database consists of about 30,000 tables
- Semantic Web applications
- Reverse engineering

A conceptual diagram with thousands of elements does not worth a lot.

The objective:

Device techniques to **aid the understanding** and the visualization of large conceptual diagrams.

E.g. generation of more abstract or focused views.
Existing Solutions for Managing Large Diagrams (1/3)

[1] ER Clustering
- The classical methods require human input
  - [Felldman & Miller'86, Teory et al. 89, Gandhi et al. 94, Campbell et al. ’96]
- Automatic methods not tested in large ER diagrams
  - [Akoka & Comyn-Wattiau’96, Raugh & Stickel’92]

Existing Solutions for Managing Large Diagrams (2/3)

[2] Graph Drawing and Abstraction
- Automatically generated layouts are not satisfying
  - Most (if not all) of the layouts are still created manually
- Hierarchical decomposition techniques that are used for visualizing big plain graphs have not been applied or tested on conceptual diagrams
Existing Solutions for Managing Large Diagrams (3/3)

[3] Visualization methods

– Not very helpful for our problem
  • Hyperbolic trees: appropriate only for trees
  • FishEye View: not really helpful and computationally hard

The Idea

Try to rank the elements of the diagram according to their “importance”
Why ranking can be useful?

- Gradual visualization (understanding): from the most important to the less
  - Provision of top-k diagrams for successive values of k

  ![Diagrams](top-1 to Real diagram)

- Other applications: Keyword searching, ...

Remark: Web Searching is an Analogous problem

- The Web graph is a very big graph too.
- Link Analysis has been proved very successful in Web searching (and recently in many other domains)
  - Main techniques: PageRank, HITS.

  ![Diagram](Define a PageRank-like scoring scheme for ER diagrams)
Applying PageRank on ER diagrams

Differences with the Web

<table>
<thead>
<tr>
<th>Web</th>
<th>ER</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Directed links</td>
<td>• Undirected relationships</td>
</tr>
<tr>
<td>• Binary links</td>
<td>• n-ary relationships</td>
</tr>
<tr>
<td>• ignore self hyperlinks</td>
<td>• cyclic relationships are important</td>
</tr>
</tbody>
</table>

Person should be ranked higher than City

Viewing an ER diagram as a Markov chain

• Entity type => state
• Binary relationship type => two counterpoising transitions
  – An n-ary relationship type is first replaced by n(n-1)/2 binary relationship types
What about ISA Hierarchies?

- We could view an ISA link as transitions (one way or two-way).

Our approach
- We ignore them, but
- We have an optional preprocessing step where we **collapse** each ISA hierarchy into one node (the root(s) of the hierarchy)
  - that collects all attributes and relationship types

Towards defining EntityRank

**Deriving the Probability Transition Matrix** $M$

**A**: Adjacency matrix

\[
\begin{array}{ccccc}
 & a & b & c & d & e \\
 a & 0 & 0 & 2 & 0 & 0 \\
b & 0 & 0 & 1 & 0 & 0 \\
c & 2 & 1 & 0 & 1 & 1 \\
d & 0 & 0 & 1 & 2 & 1 \\
e & 0 & 0 & 1 & 1 & 0 \\
\end{array}
\]

By normalizing each row of $A$ to sum to 1

**M**: transition matrix

\[
\begin{array}{ccccc}
 & a & b & c & d & e \\
 a & 0 & 0 & 1 & 0 & 0 \\
b & 0 & 0 & 1 & 0 & 0 \\
c & 2/5 & 1/5 & 0 & 1/5 & 1/5 \\
d & 0 & 0 & 1/4 & 2/4 & 1/4 \\
e & 0 & 0 & 1/2 & 1/2 & 0 \\
\end{array}
\]
EntityRank

- Suppose a random E-R surfer who at each time step is at some entity type e and then:
  - with probability \( q \) (e.g. \( q=0.15 \)) jumps to a randomly picked entity type, and
  - with probability \( (1-q) \) jumps to an entity type that is connected with the e

- EntityRank score := stationary probability

- This process defines a Markov chain with transition matrix:
  - \( q \cdot U + (1-q) \cdot M \)
  - where \( U[e_i,e_j]=1/N \) for all \( i,j \), where \( N \) the number of entity types
  - As the transition graph is strongly connected and non-bipartite, the fundamental theorem of Markov chains apply.
  - The score matrix is the principal right eigenvector of the following transition matrix:
    \((q \cdot U + (1-q) \cdot M)^T\)

EntityRank (II)

- The matrix equation gives the following equation for each entity type e:

\[
Sc(e) = \frac{q}{N} + (1-q) \sum_{e' \in conn(e)} \frac{Sc(e')}{|conn(e')|}
\]

\(conn(e)\): bag (duplicates allowed) with all entities types that are connected with e
**B(iased)EntityRank**

- The probability of jumping to a random etype is not the same for all, but it depends on the number of its attributes.

- This process defines a Markov chain with transition matrix:
  \[ q \begin{pmatrix} B & (1-q) \end{pmatrix} \cdot M \]  where  \( B_{i,j} = \frac{\text{attrs}(i)}{\text{all attrs}} \)

\[ S(e) = q \frac{|\text{attrs}(e)|}{|\text{all attrs}|} + (1-q) \sum_{e' \in \text{conn}(e)} \frac{S(e')}{|\text{conn}(e')|} \]

- **BEntityRank** is a well founded method for incorporating external domain knowledge, and preferences:
  - Number of tuples in the associated database tables
  - Application program’s call trees
  - User feedback while interacting with top-k diagrams

---

**Evaluating Ranking Methods for Conceptual Diagrams**

Evaluation is difficult. The most safe evaluation is to test it on well known schemas

- **Empirical**

- **Formal Experimental (TREC/INEX-like)**
  - Derive the Ideal Ordering of the elements of a schema
    - by aggregating the top-k lists provided by several experts on that schema
  - Compare an automatically derived ordering w.r.t. the ideal ordering
    - metrics
      - k-precision, ....
Evaluating Ranking Methods for Conceptual Diagrams

Typical Distribution of Scores

![Graph showing typical distribution of scores.](image)

Large ER diagrams tend to have a well-connected kernel

Zipf’s Law

Evaluation on DICO1

Top-3 diagram of
Evaluation on the TELEBIB schema

Belgian national standard for exchanging messages between insurance companies (339 entity types and 232 relationship types)
End of Part A (Deriving top-k ER Diagrams (using Link Analysis))

• The diagrams of large schemas are very difficult to understand and visualize

• **Ranking** can be very helpful
  – *Top-k diagrams* that are derived using link analysis can aid the understanding, the visualization, and the drawing of such diagrams.

• Ranking using EntityRank and BEntityRank seems to be successful
  – This is the first work that exploits LA techniques for ER diagrams
  – Only ER clustering is somehow related, but
    • ER clustering techniques are mainly manual
    • The automatic ones have not been tested on big ER diagrams so their effectiveness is unknown
    • Anyway, they can be considered as complementary approaches

Part B: Drawing ER diagrams
How to visualise these top-k diagrams?

• For drawing automatically the top-k diagrams we combine the spring-model algorithm with the magnetic-spring model in a way that is appropriate for ER diagrams.

• We view an ER diagram as a mechanical system
  – Force model A
  – Force model B

• Drawing algorithm
  – Algorithm that simulates the mechanical system
  – Its seeks for a configuration with locally minimal energy
Force Model A

Force Model A: Computing the exerted forces

- Force exerted on an entity type:

\[
F(e_i) = \sum_{e_j \in \text{CON}(e_i)} f(e_j, e_i) + \sum_{e_j \in \text{E}, e_i \neq e_j} g(e_j, e_i) + \sum_{e_j \in \text{CONISA}(e_i)} h(e_j, e_i)
\]

- **String force** from connections
- **Electrical repulsion** from every other particle
- **Magnetic (rotational) force** from ISA connections
Force Model A: Configuration parameters

\[
F(e_i) = \sum_{e_j \in \text{CON}(e_i)} f(e_j, e_i) + \sum_{e_j \in \text{E}, e_i \neq e_j} g(e_j, e_i) + \sum_{e_j \in \text{CONISA}(e_i)} h(e_j, e_i)
\]

\[
f_x(e_i) = \sum_{e_j \in \text{CON}(e_i)} K_{i,j}^s \left(d(p_i, p_j) - L_{i,j}\right) \frac{x_j - x_i}{d(p_i, p_j)}
\]

\[
g_x(e_i) = \sum_{e_j \in \text{E}, e_i \neq e_j} K_{i,j}^t \frac{x_i - x_j}{d(p_i, p_j)^2 d(p_i, p_j)}
\]

\[
h_x(e_i) = \sum_{e_j \in \text{CONSUP}(e_i)} K^m \frac{x_j - x_i}{L_{i,j}} + \sum_{e_j \in \text{CONSUB}(e_i)} K^m \frac{x_j - x_i}{L_{i,j}}
\]

The role of the forces \(f, g,\) and \(h\) and of the configuration parameters
Force Model B

- Relationship types are also considered as particles in order to discourage overlaps.

B: Drawing ER diagrams
Experimental Evaluation> Multiple Isa Hierarchies
Experimental Evaluation
FM A vs FM B

- FM A
- FM B
  - The tentangles of binary relationship types are unnecessarily not aligned.
  - Computationally more expensive

On bigger diagrams

Artificial diagram
Real ER diagram
Experimental Evaluation> On real diagrams
Top-5: FM A (after 2 different initial placements)

Experimental Evaluation> On real diagrams
Top-5: FM B
Experimental Evaluation> On real diagrams
Top-11: FM B (dense diagram)

Solutions:
- Play with the configuration parameters
  - local density-based configuration of parameters (future research)
- or just scale up the entire diagram

Less stiff springs

Higher electrical repulsion
Experimental Evaluation> On real diagrams
Top-11: After manual rectification/beautification

How to Tame a Very Large ER Diagram
(using Link Analysis and Force-Directed Placement Algorithms)

Concluding Remarks

• The diagrams of large schemas are very difficult to understand and visualize

• Top-k diagrams that are derived using link analysis can aid the understanding, the visualization, and the drawing of such diagrams.

• The experiments on real diagrams were successful

• Automatic drawing algorithms can be applied on top-k diagrams and give satisfying layouts (at least for small k)
Future Research

• Ranking
  – BEntityRank and User Feedback
  – Test Collections for Evaluations
  – Take into account multiplicity constraints?
  – Application on other kinds of diagrams
    • Relational schemas, SWeb ontologies, Social Networks, ...

• Drawing
  – Local-Density Adaptation (EntityRank scores might help)