INTEGRATING CONNECTION SEARCH IN GRAPH QUERIES

Angelos C. Anadiotis, Ioana Manolescu, Madhulika Mohanty

Inria & IPP

ICDE 2023
Anaheim, CA April 3 – 7, 2023
MOTIVATING EXAMPLE
How are the US entrepreneurs, French entrepreneurs and French politicians related?
How are the US entrepreneurs, French entrepreneurs and French politicians related?

Connecting trees. Doug, Falcon and Carole are leaves.
# State of the Art

<table>
<thead>
<tr>
<th>Requirements/Existing</th>
<th>Exact match</th>
<th>(Label-constrained) regular paths between any two nodes</th>
<th>Connecting tree search</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPARQL</td>
<td>✓ (US entrepreneurs, French politicians, French entrepreneurs)</td>
<td>✓ (check but not return) $x$ – knows*–&gt; $y$</td>
<td>✗</td>
</tr>
<tr>
<td>Cypher/GQL</td>
<td>✓</td>
<td>✓ $x$-[*]-$y$</td>
<td>✗</td>
</tr>
<tr>
<td>Keyword Search Algorithms (BANKS, BLINKS, DBXplorer, etc.)</td>
<td>✗</td>
<td>✗</td>
<td>✓ (based on keywords alone, various pruning strategies)</td>
</tr>
</tbody>
</table>

How are the US entrepreneurs, French entrepreneurs and French politicians related?
# State of the Art

<table>
<thead>
<tr>
<th>Requirements/Existing</th>
<th>Exact match</th>
<th>(Label-constrained) regular paths between any two nodes</th>
<th>Connecting tree search</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPARQL</td>
<td>✓ (US entrepreneurs, French politicians, French entrepreneurs)</td>
<td>✓ (check but not return)</td>
<td>✗</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$x \text{ – knows}^* \rightarrow y$</td>
<td></td>
</tr>
<tr>
<td>Cypher/GQL</td>
<td>✓</td>
<td>✓ $x-[^*]-y$</td>
<td>✗</td>
</tr>
<tr>
<td>Keyword Search Algorithms (BANKS, BLINKS, DBXplorer, etc.)</td>
<td>✗</td>
<td>✗</td>
<td>✓ (based on keywords alone, various pruning strategies)</td>
</tr>
</tbody>
</table>

Keyword Search has high complexity (Group Steiner Tree – NP-Hard)

How are the US entrepreneurs, French entrepreneurs and French politicians related?
How are the US entrepreneurs, French entrepreneurs and French politicians related?

**(P1)** A query language supporting such queries
How are the US entrepreneurs, French entrepreneurs and French politicians related?

(P1) A query language supporting such queries

(P2) General tree search
  - Undirected search.
  - Find all answers (under space and time budget).
  - Independent of the cost function.
How are the US entrepreneurs, French entrepreneurs and French politicians related?

(P1) A query language supporting such queries

(P2) General tree search
  - Undirected search.
  - Find all answers (under space and time budget).
  - Independent of the cost function.

(P3) Efficient execution algorithms
Support for Extended Queries (EQ)

Given a graph and set of node/edge properties, GPML supports:

- **Path Patterns (PPs)** of the form:
  
  MATCH
  
  \( (v: \text{Alice} \text{WHERE } v.\text{type}= \text{entrepreneur}) \) \( -[e: \text{citizenOf}] \rightarrow \) \( (w \text{WHERE } w.\text{type}= \text{country}) \)

- PPs can contain Regular Paths:
  
  MATCH \( p = (x) \) \(-[y: \text{founded}] \rightarrow \ast(z)\)

- **Graph Patterns (GPs)**
  
  Conjunction of PPs
**SUPPORT FOR EXTENDED QUERIES (EQ)**

Given a graph and set of node/edge properties, GPML supports:

- **Path Patterns (PPs)** of the form:
  
  MATCH 
  
  (v: Alice WHERE v.type=entrepreneur) 
  −[e: citizenOf]→ 
  (w WHERE w.type=country)

- PPs can contain Regular Paths:
  
  MATCH p = (x) −[y: founded]→*(z)

- **Graph Patterns (GPs)**
  
  Conjunction of PPs

Our extension: **Connecting Tree Patterns (CTPs)**

- \( n \) input variables, 1 output variable \((x, y, z, w)\)

Integrating Connection Search in Graph Queries, ICDE 2023

April 06, 2023
How are the US entrepreneurs, French entrepreneurs and French politicians related?

MATCH
(x WHERE x.type = entrepreneur) −[a: citizenOf]→ (b: USA),
(y WHERE y.type= entrepreneur) −[c: citizenOf]→ (d: France),
(z WHERE z.type = politician) −[e: citizenOf]→ (f: France),
(x, y, z, w)
RETURN w;
**QUERY SEMANTICS (GP)**

MATCH
(x WHERE x.type = entrepreneur) −[a: citizenOf]→ (b: USA),
(y WHERE y.type = entrepreneur) −[c: citizenOf]→ (d: France),
(z WHERE z.type = politician) −[e: citizenOf]→ (f: France),
(x, y, z, w)
RETURN w;

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>Alice</td>
<td>Elon</td>
</tr>
<tr>
<td>Carole</td>
<td>Doug</td>
<td>Falcon</td>
</tr>
</tbody>
</table>
**QUERY SEMANTICS (CTP)**

MATCH
(x WHERE x.type = entrepreneur) −[a: citizenOf]→ (b: USA),
(y WHERE y.type= entrepreneur) −[c: citizenOf]→ (d: France),
(z WHERE z.type = politician) −[e: citizenOf]→ (f: France),
(x, y, z, w)
RETURN w;

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>Alice</td>
<td>Elon</td>
<td></td>
</tr>
<tr>
<td>Carole</td>
<td>Doug</td>
<td>Falcon</td>
<td></td>
</tr>
<tr>
<td>Carole</td>
<td>Alice</td>
<td>Falcon</td>
<td>....</td>
</tr>
</tbody>
</table>

and many more
MATCH
(x WHERE x.type = entrepreneur) −[a: citizenOf]→ (b: USA),
(y WHERE y.type= entrepreneur) −[c: citizenOf]→ (d: France),
(z WHERE z.type = politician) −[e: citizenOf]→ (f: France),
(x, y, z, w)
RETURN w;

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>Alice</td>
<td>Elon</td>
<td></td>
</tr>
<tr>
<td>Carole</td>
<td>Doug</td>
<td>Falcon</td>
<td></td>
</tr>
<tr>
<td>Carole</td>
<td>Alice</td>
<td>Falcon</td>
<td></td>
</tr>
<tr>
<td>and many more</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**GAM ALGORITHM**

Graph integration of structured, semistructured and unstructured data for data journalism, A. Anadiotis et al. Information Systems J. (2022)

Build rooted trees:

- **INIT(n)**
  - Carole $t_1$
  - Bob $t_4$
  - Falcon $t_2$
  - Elon $t_5$
  - Doug $t_3$
  - Alice $t_6$
**GAM ALGORITHM**  
Graph integration of structured, semistructured and unstructured data for data journalism, A. Anadiotis et al. Information Systems J. (2022)

Build **rooted trees**:

- **INIT(n)**

- **GROW(t, e)**

- **MERGE (t, t’)**
Further:

- **Property:** Complete
- Minimal answers *only*.
- Builds all rooted trees. ×
OPTIMIZATION 1: EDGE SET PRUNING (ESP)

- Keep only the first rooted tree built for the same set of edges.
OPTIMIZATION 1: EDGE SET PRUNING (ESP)

- Keep only the first rooted tree built for the same set of edges.

- **Property**: Complete for 2-input CTPs. ✓
OPTIMIZATION 1: EDGE SET PRUNING (ESP)

- **Property**: Complete for 2-input CTPs. ✓

- Incomplete for a CTP of arity ≥3. ✗

- Example:
  - Connection between Alice, Bob and Falcon

---

Keep only the first rooted tree built for the same set of edges.
OPTIMIZATION 2: MERGE-ORIENTED ESP (MOESP)

- Build MoESP trees from new tree $t$ as follows:
  - $t'$ has the same edges (and nodes) as $t$, but
  - $t'$ is rooted in a seed $r$, distinct from the root of $t$
**OPTIMIZATION 2: MERGE-ORIENTED ESP (MOESP)**

- Build MoESP trees from new tree $t$ as follows:
  - $t'$ has the same edges (and nodes) as $t$, but
  - $t'$ is rooted in a seed $r$, distinct from the root of $t$

- **Property:** Finds all path results. ✓
- May still fail to find some results. ✗
OPTIMIZATION 3: LIMITED ESP (LESP)

- Signature (s) for each node *n*:
  - #Seed-sets having paths from seeds to *n*.

- Limit ESP when the root:
  - Has *s*>=3.
  - Has 3 or more adjacent edges.
Variant with ESP, MoESP (inject more trees) and LESP (limit pruning on some Merges).

- **Property:** Complete for CTPs of arity 3. ✓

Still incomplete for some CTPs of arity >= 4.

- **Property:** Identified class of solutions guaranteed to be found for CTPs of any arity (refer to the paper).
  - The frequent cases covered. ✓
SCALABILITY ON BARABASI ALBERT GRAPHS

Timeout 25 minutes.
SCALABILITY ON BARABASI ALBERT GRAPHS

MoLESP scales well with the size of Barabasi-Albert graphs.
SCALABILITY ON BARABASI ALBERT GRAPHS

![Graph Scalability Chart](image)

- GAM
- MoLESP
- MoESP
- LESP
- ESP

$m=3$
MoLESP is 2x faster than GAM, slightly slower than MoESP.
RECALL ON BARABASI ALBERT GRAPHS
MoLESP has a perfect recall even for $m=4$. 

**RECALL ON BARABASI ALBERT GRAPHS**
COMPARISON WITH KEYWORD SEARCH

SOTA: QGSTP


Dataset: DBPedia
#Queries: 312

Timeout 15 minutes
**COMPARISON WITH KEYWORD SEARCH**

**SOTA: QGSTP**

Dataset: DBPedia

#Queries: 312

MoLESP is 4x faster in finding result.

COMPARISON WITH GQ ENGINES

Timeout 15 minutes


Integrating Connection Search in Graph Queries, ICDE 2023
COMPARISON WITH GQ ENGINES

MoLESP is the only feasible path returning algorithm.


Integrating Connection Search in Graph Queries, ICDE 2023
COMPARISON WITH GQ ENGINES
PATH STITCHING

MoLESP scales well with the graph size and the cost of adding CTP is minimal.

CONCLUSION

- Extension to GPML by using CTPs.
  - Supports asking for connecting trees.

- MoLESP: Efficient search algorithm for the connecting trees.

- Future work:
  - Smart execution strategies for jointly optimizing GPs and CTPs.
  - Optimized execution of multiple CTPs.
THANK YOU

Project Web site: https://team.inria.fr/cedar/connectionlens/

Code and datasets for this paper: https://gitlab.inria.fr/cedar/extended-graph-querying