DDMCE : Distributable Dynamic Maximal Clique Exploration in Large Scale Graphs

Ovidiu Şerban*,*** Alexandre Pauchet*, Alexandrina Rogozan*, Jean-Pierre Pécuchet*

LITIS Laboratory, INSA Rouen
Saint-Etienne-du-Rouvray, France
{firstname.surname}@insa-rouen.fr

*Department of Computer Science
Faculty of Mathematics and Computer Science
"Babeş-Bolyai" University,
Cluj-Napoca, Romania

18 October 2012
Sample subtitle from the "The Big Bang Theory" TV Series

1. Good evening.
2. I’m your guest lecturer, Dr. Sheldon Cooper.
3. I was expecting applause,
4. but I suppose stunned silence is equally appropriate.
5. I agreed to speak to you this evening,
6. because I was told that you’re the best and the brightest
7. of this university’s doctoral candidates.
... to structured data.
Contextonym

- Introduced by Ji et al. [4] and represent a strong link between the words, structured in a graph
- A contextonym, in graph theory is a maximal clique
- Ji et al. propose also a couple of global and localised filtering techniques in order to reduce the data

Cliques

- A clique is complete sub-graph, which means that every node in the structure is connected to the others
- $\forall q \in \{\text{maximalClique}\} \iff \nexists q' \in \{\text{Clique}\} \Rightarrow q \subset q'$
- The algorithm for clique extraction is also referred as Maximal Clique Exploration (MCE)
Algorithmic constraints

- Suitable for large data exploration
- Parallel and(or) distributed
- Suitable for dynamic data
Algorithmic constraints

- Suitable for large data exploration
- Parallel and(or) distributed
- Suitable for dynamic data

Current approaches

- Most of the current exploratory algorithms are extensions of Bron-Kerbosch Algorithm [1]
- Koch [5], Tomita et al. [10] and Cazals et al. [2] provided pivot selection strategies to improve the algorithm
- Dynamic data: Stix [9]
- Distributed: Pardolos et al. [7] and Schmidt et al. [8]
- Google Pregel [6] - a platform for graph-processing
is decomposed in

Q: \{\{2,1,5\}, \{2,4,3\}, \{6,4,5\}\}
DDMCE - Design

Queue Generator

Message Queue

P_1

\ldots

P_n

sT_1

\ldots

sT_k

Ovidiu Şerban (INSA Rouen and UBB) JFGG’12 Presentation, October 2012

October 2012 9 / 27
Node exploration

function explore(P, D, vi)
    if P = ∅ then
        if D = ∅ then
            markFinal(vi)
        end if
        treeCleanup(vi)
    else
        up ← |choosePivot|(P)
        for all v ∈ P \ N(up) do
            ve ← |createNode|(v)
            P_v ← P ∩ N(v)
            D_v ← D ∩ N(v)
            |createExploreMessage|(P_v, D_v, ve)
            D ← D ∪ {v}
            P ← P \ {v}
        end for
    end if
end function

Queue Generator

function Generator(P,Q)
    up ← |choosePivot|(P)
    for all v ∈ P \ N(up) do
        ve ← |createNode|(v)
        P_v ← P ∩ N(v)
        D_v ← D ∩ N(v)
        |push|(Q, P_v, D_v, ve)
        D ← D ∪ {v}
        P ← P \ {v}
    end for
end function
**Pivot selection**

**Require:** $P$ - potential node set

**Ensure:** pivot, according to Tomita et al.[10] Strategy

function choosePivot($P$)

\[
pivot \leftarrow \max_{u \in P} |P \cap N(u)|
\]

end function

is decomposed in

\[
Q: \{\{2,1,5\}, \{2,4,3\}, \{2,4,5\}, \{6,4,5\}\}
\]
\[ P = \{1, 2, 3, 4, 5, 6\} \]
\[ D = \emptyset \]
P = \{1, 2, 3, 4, 5, 6\}
D = \emptyset
Pivot = 2
\[ P = \{1, 3, 4, 5\} \]
\[ D = \emptyset \]
\[ \text{Pivot} = 4 \]
P = \{5\}
D = \emptyset
Pivot = null
P = ∅
D = ∅
Pivot = null
\(P = \{3, 5\}\)
\(D = \{1\}\)
Pivot = null
DDMCE - Cleanup

[Tree diagram with nodes labeled 1, 2, 4, 5, 3, 6, 5. Node 4 is highlighted, and there is an arrow indicating a move from node 1 to node 2.]
DDMCE - Distribution

Ovidiu Şerban (INSA Rouen and UBB)  JFGG’12 Presentation, October 2012
DDMCE - Dynamic case

Edge addition

Edge addition - Pivot influence
Technical data

- We used Java programming language (Oracle distribution, version 1.7.0 for Linux x64 platforms)
- Colt library [3] (version 1.2.0), for bit sets operations
- All the sets and graph structures are changing their representation from dense (bit vectors) to sparse (hash maps) representation, where needed
- A Xeon machine was used to test the dynamic part of the experiments, having 4 cores at 1.8 Ghz per core and 3 Gb of RAM available for user mode
<table>
<thead>
<tr>
<th>n</th>
<th>ρ</th>
<th>c</th>
<th>Static</th>
<th>Dynamic</th>
<th>Dynamic/Static</th>
<th>n</th>
<th>ρ</th>
<th>c</th>
<th>Static</th>
<th>Dynamic</th>
<th>Dynamic/Static</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.01</td>
<td>1</td>
<td>.342</td>
<td>.029</td>
<td>.084</td>
<td></td>
<td>.01</td>
<td>1</td>
<td>1.687</td>
<td>.125</td>
<td>.074</td>
<td></td>
</tr>
<tr>
<td>.01</td>
<td>2</td>
<td>.250</td>
<td>.021</td>
<td>.086</td>
<td></td>
<td>.01</td>
<td>2</td>
<td>1.324</td>
<td>.100</td>
<td>.076</td>
<td></td>
</tr>
<tr>
<td>.01</td>
<td>4</td>
<td>.240</td>
<td>.022</td>
<td>.092</td>
<td></td>
<td>.01</td>
<td>4</td>
<td>5.673</td>
<td>.449</td>
<td>.079</td>
<td></td>
</tr>
<tr>
<td>.03</td>
<td>1</td>
<td>.598</td>
<td>.061</td>
<td>.102</td>
<td></td>
<td>.03</td>
<td>1</td>
<td>30.805</td>
<td>1.267</td>
<td>.041</td>
<td></td>
</tr>
<tr>
<td>.03</td>
<td>2</td>
<td>.557</td>
<td>.043</td>
<td>.077</td>
<td></td>
<td>.03</td>
<td>2</td>
<td>18.585</td>
<td>1.103</td>
<td>.059</td>
<td></td>
</tr>
<tr>
<td>.03</td>
<td>4</td>
<td>.543</td>
<td>.043</td>
<td>.078</td>
<td></td>
<td>.03</td>
<td>4</td>
<td>17.828</td>
<td>1.102</td>
<td>.062</td>
<td></td>
</tr>
<tr>
<td>.05</td>
<td>1</td>
<td>.776</td>
<td>.119</td>
<td>.154</td>
<td></td>
<td>.05</td>
<td>1</td>
<td>90.080</td>
<td>3.854</td>
<td>.043</td>
<td></td>
</tr>
<tr>
<td>.05</td>
<td>2</td>
<td>.836</td>
<td>.098</td>
<td>.117</td>
<td></td>
<td>.05</td>
<td>2</td>
<td>54.499</td>
<td>2.804</td>
<td>.051</td>
<td></td>
</tr>
<tr>
<td>.05</td>
<td>4</td>
<td>.996</td>
<td>.101</td>
<td>.149</td>
<td></td>
<td>.05</td>
<td>4</td>
<td>52.718</td>
<td>2.746</td>
<td>.052</td>
<td></td>
</tr>
<tr>
<td>.07</td>
<td>1</td>
<td>.992</td>
<td>.210</td>
<td>.212</td>
<td></td>
<td>.07</td>
<td>1</td>
<td>408.919</td>
<td>19.324</td>
<td>.047</td>
<td></td>
</tr>
<tr>
<td>.07</td>
<td>2</td>
<td>.886</td>
<td>.164</td>
<td>.185</td>
<td></td>
<td>.07</td>
<td>2</td>
<td>54.499</td>
<td>2.804</td>
<td>.051</td>
<td></td>
</tr>
<tr>
<td>.07</td>
<td>4</td>
<td>1.071</td>
<td>.160</td>
<td>.149</td>
<td></td>
<td>.07</td>
<td>4</td>
<td>52.718</td>
<td>2.746</td>
<td>.052</td>
<td></td>
</tr>
<tr>
<td>.10</td>
<td>1</td>
<td>1.617</td>
<td>.483</td>
<td>.298</td>
<td></td>
<td>.10</td>
<td>1</td>
<td>408.919</td>
<td>19.324</td>
<td>.047</td>
<td></td>
</tr>
<tr>
<td>.10</td>
<td>2</td>
<td>1.215</td>
<td>.388</td>
<td>.319</td>
<td></td>
<td>.10</td>
<td>2</td>
<td>245.741</td>
<td>11.875</td>
<td>.048</td>
<td></td>
</tr>
<tr>
<td>.10</td>
<td>4</td>
<td>1.433</td>
<td>.365</td>
<td>.255</td>
<td></td>
<td>.10</td>
<td>4</td>
<td>235.444</td>
<td>11.580</td>
<td>.049</td>
<td></td>
</tr>
<tr>
<td>5,000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.01</td>
<td>1</td>
<td>1.687</td>
<td>.125</td>
<td>.074</td>
<td></td>
<td>.01</td>
<td>1</td>
<td>13.114</td>
<td>2.92</td>
<td>.022</td>
<td></td>
</tr>
<tr>
<td>.01</td>
<td>2</td>
<td>1.266</td>
<td>.099</td>
<td>.078</td>
<td></td>
<td>.01</td>
<td>2</td>
<td>9.246</td>
<td>1.94</td>
<td>.021</td>
<td></td>
</tr>
<tr>
<td>.01</td>
<td>4</td>
<td>1.324</td>
<td>.100</td>
<td>.076</td>
<td></td>
<td>.01</td>
<td>4</td>
<td>9.010</td>
<td>.541</td>
<td>.060</td>
<td></td>
</tr>
<tr>
<td>.03</td>
<td>1</td>
<td>9.010</td>
<td>.541</td>
<td>.060</td>
<td></td>
<td>.03</td>
<td>1</td>
<td>54.499</td>
<td>2.804</td>
<td>.051</td>
<td></td>
</tr>
<tr>
<td>.03</td>
<td>2</td>
<td>5.673</td>
<td>.449</td>
<td>.079</td>
<td></td>
<td>.03</td>
<td>2</td>
<td>52.718</td>
<td>2.746</td>
<td>.052</td>
<td></td>
</tr>
<tr>
<td>.03</td>
<td>4</td>
<td>5.363</td>
<td>.432</td>
<td>.081</td>
<td></td>
<td>.03</td>
<td>4</td>
<td>52.718</td>
<td>2.746</td>
<td>.052</td>
<td></td>
</tr>
</tbody>
</table>
DDMCE - Results: 1,000 nodes

- $\rho = 1\%$
- $\rho = 3\%$
- $\rho = 25\%$
- $\rho = 30\%$
DDMCE - Results: 5,000 nodes

ρ = 1%

ρ = 3%

ρ = 7%

ρ = 10%
DDMCE - Results: 10,000 nodes

\[ \rho = 1\% \]

\[ \rho = 3\% \]

\[ \rho = 7\% \]

\[ \rho = 10\% \]
Conclusion and Future work

**Conclusion**
- DDMCE offers interesting results for dynamic processing
- For large data, the differences between serial and parallel implementation, are significant
- Due to its design, DDMCE assures a graph reduction on every exploration step

**Future work**
- Migrate to a fully distributed algorithm
- Offer a better balancing for processing, for extreme cases
Merci pour votre attention !
C. Bron and J. Kerbosch.  
Algorithm 457: finding all cliques of an undirected graph.  

F. Cazals and C. Karande.  
A note on the problem of reporting maximal cliques.  

CERN.  
The colt distribution: Open source libraries for high performance scientific and technical computing in java, April 2012.  
CERN - European Organization for Nuclear Research.

Hyungsuk Ji, Sabine Ploux, and Eric Wehrli.  
Lexical knowledge representation with contextonyms.  

I. Koch.  
Enumerating all connected maximal common subgraphs in two graphs.  

Pregel: a system for large-scale graph processing.

P.M. Pardalos, J. Rappe, and M.G.C. Resende.
An exact parallel algorithm for the maximum clique problem.

A scalable, parallel algorithm for maximal clique enumeration.

V. Stix.
Finding all maximal cliques in dynamic graphs.

E. Tomita, A. Tanaka, and H. Takahashi.
The worst-case time complexity for generating all maximal cliques and computational experiments.