Applying MapReduce principle to High level information fusion

Claire Laudy, Johann Dreo, Christophe Gouguenheim

Applying MapReduce principle to High level information fusion

Claire Laudy
Thales Research & Technology
claire.laudy@thalesgroup.com

Johann Dreo
Thales Research & Technology
johann.dreo@thalesgroup.com

Christophe Gouguenheim
Thales Research & Technology
christophe.gouguenheim@thalesgroup.com

***Abstract*—The InSyTo Synthesis framework is based on graph structures, graph algorithms and similarity measures for soft data fusion managing inconsistencies. The framework can be used to enable non-redundant additions to an information network, as well as graph based information query on several applications. The graph fusion algorithm relies on the search of a *maximum common subgraph* isomorphism, which makes it a difficult problem, especially on large graphs. In this work, the subgraph matching algorithm is partially parallelized, based on the MapReduce approach and on the Hadoop framework. Using Hadoop enables the management of big graphs, first by avoiding the load of the graphs in memory and secondly by distributing the computations over several processing nodes. Our experiments on the Global Terrorism Database (which contains the descriptions of more than 113,000 terrorist attacks in a graph of more than 20,000,000 nodes) shows that InSyTo Synthesis now scales to so-called "big data" applications.**

I. Introduction

Soft data fusion is an ever growing trend in the information fusion community. More and more tracks dedicated to soft information are organized within the International Conference on Information Fusion over the years and numerous authors stress the need for soft data management and fusion. For instance in the detection of people and complex activities, the use of soft information sources is critical. The authors of [1] describe a 5 year program involving several academic actors that aim at addressing major stakes of soft data fusion. It includes the development of a framework as well as evaluation methods. In addition, many authors relate about new issues raised by soft data fusion. Among them [2] and [3] quote natural language processing, transformation of data into comprehensive and semantic data structures, soft data association and graph matching. Studies such as [4] and [5] emphasize on the importance of integrating soft data in situation awareness support systems. As the automation of soft data fusion is a very challenging issue, due to, e.g., error estimation, normalization and context extraction for information interpretation, the authors propose a mixed approach that embeds the participation of a human analyst to the fusion process.

Within their day to day work, intelligence analysts pile up a large amount of information. The different information items are linked through a big information network, as the analysts express part of their reasoning and domain knowledge through links between different pieces of information, piled up within one or more working sessions. One of the issue for using such an amount of information is to be able to access relevant parts of it efficiently. For example, this enables highlighting schemes of actions of terrorist groups, recognize modus operandi and deviation from usual behavior for instance.

We previously developed a framework based on graph structures, graph algorithm and similarity measures for soft data fusion managing inconsistencies (InSyTo Synthesis v1 [6], [7]). The use case presented in [8] dealt with the management of an information network containing descriptions of entities (companies, universities...) that collaborate through several media such as research project, scientific papers and so on. The InSyTo Synthesis framework was used to enable non-redundant information addition to the information network, as well as graph based information query. This network contains more than 10000 nodes and the operations took a few seconds. The processing time was operationally accepted and the memory load was not an issue. However, the new step envisioned for our work on soft data fusion, was to enable the management of bigger graphs.

The graph fusion algorithm relies on the search for matches between sub-graphs and more precisely for a maximal matching subgraph. Maximal subgraph matching is used in order to determine where to add information in an information graph, and which parts of two information graphs are redundant and should thus be fused rather than be repeated twice in the graph resulting from the fusion.

The decision problem of whether two *graphs* match is well studied and is in NP [9]. But the problem of finding matching *subgraphs*, known as the *subgraph isomorphism* problem, is known to be NP-complete and difficult to solve in parallel [10]. This problem is well studied and have leds to a lot of algorithms, among which several have been parallelized [10], [11], [12]. The most difficult graph matching problem in our case is to find the set of subgraphs that *maximize* the matching. This *maximum common subgraph* isomorphism problem (MCS) is known to be NP-hard [13], which makes it intractable on large graphs [14]. It is thus often solved with tools from the combinatorial optimization domain [15], [16] like constraint programming [17], [18], but the parallelization of MCS algorithms remains to be studied.

The InSyTo Synthesis fusion algorithm relies on subgraph isomorphism and maximum subgraph isomorphism algorithms. Both those algorithms must solve highly combinatoric problems, on which the execution time may become too long to satisfy user requirements. To manage this problem, we present a new version of the InSyTo Synthesis framework in which the subgraph isomorphism algorithm is partially parallelized. Our approach is based on the MapReduce principle and

the implementation use the Hadoop framework [19]. Using Hadoop enables the management of big graphs by avoiding the load of the graphs in memory and distributing the computations over several processing nodes.

The paper is organized as follows. The second section is dedicated to the presentation of the soft data fusion algorithms. We present the approach that relies on the use of conceptual graphs formalism to express the information graphs. We also present the limitations of the first version of the soft data fusion framework. The third section presents the parallelization of those algorithms, which permits the management of bigger data graphs. After a short presentation of the Hadoop framework parts on which our work relies, we describe the MapReduce version of a subgraph matching algorithm. Section four is dedicated to our experimentations. We used the MapReduce version of the framework on a huge database containing the descriptions of more than 113,000 terrorist attacks over the world. We show how InSyTo Synthesis now scales to so-called "big data" applications. In the last section, we conclude and present future work.

II. Graph based Information Representation

A. Information fusion, information synthesis and information Query

The InSyTo Synthesis soft data fusion framework relies on the use of semantic graph structures to store soft data and uses a graph algorithm to carry out the fusion process. It enables three different operations on networks of information that are depicted on figure 1 and described hereafter.

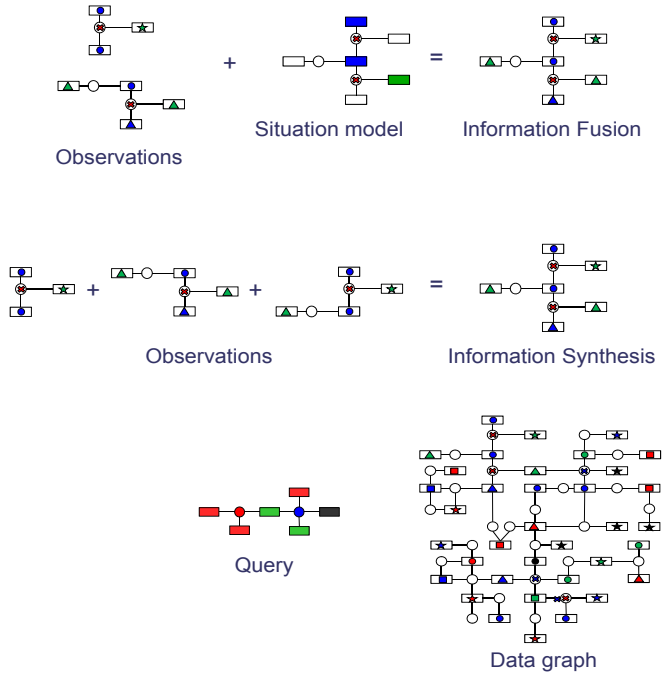

Fig. 1. InSyTo three functions

When a model of a situation of interest (e.g. a terrorist attack in a specific city at a specific date) is available, one may want to monitor the situation and raise alarms if an instance of such a situation is happening. Therefore, different observations, coming potentially from different sources, are filtered out in order to keep observations of interest only. They are then assembled through **information fusion** in order to provide a representation of the ongoing situation of interest, as precise as possible.

Information synthesis enables one to collect and organize information about a specific subject. Through information synthesis, all the gathered information items are organized into a network. The redundant part of the informations items are detected and eliminated.

All the instances of information corresponding to a specified graph pattern may be found within a network of information, through the **information query** function. In the following paper, we focus on the information query function. As we will see in section IV, given an already built information graph, our aim is to support intelligence analysts by providing means to efficiently query this information graph.

B. A graph based information representation

Graph based representations appear to be naturally well adapted to soft data. Our approach relies on the use of bipartite graphs, more specifically a subset of the conceptual graphs ([20], [21]) to represent soft data and knowledge. The conceptual graphs formalism is a model that encompasses a basic ontology (called *vocabulary*), graph structures and operations on the graphs. The vocabulary defines the different types of concepts and relations that exist in the modeled application domain, while the graphs provide a representation of the observations which are provided by the information sources.

Basic **conceptual graphs** are bipartite graphs containing concept and relation nodes. Figure 2 gives an example of a conceptual graph. The rectangular boxes represent concept nodes and the ovals represent relation nodes.

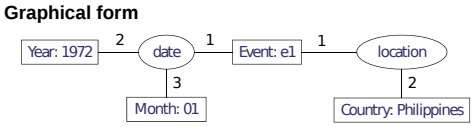

Fig. 2. Example of a conceptual graph

The term **concept** is used to refer to a concept node. The concepts represent the "things" or entities that exist. A concept is labeled with two components: the conceptual type and the individual marker.

The **conceptual type** defines the category to which the entity belongs. For instance, in Figure 2 the concept `[Country:Philippines]` is an instance of the category `Country`, i.e., its conceptual type is `Country`.

The **individual marker** relates a concept to a specific object of the world. The object represented by `[Country:Philippines]` has the name (or value) `Philippines`. The individual markers may also be undefined. An undefined or generic individual marker is either blank

or noted with a star $\star$, if the individual object referred to is unknown.

The term **relation** is used to refer to a relation node. The relation nodes of a conceptual graph indicate the relations that hold between the different entities of the situation that is represented. Each relation node is labeled with a relation type that points out the kind of relation that is represented.

The notion of **vocabulary** was defined in [21]. The concept types and the conceptual relation types, which are used to label the concept and relation nodes, are organized in hierarchies.

We restrict our approach to relation types that are unordered in order to manage only one hierarchy, the concept types hierarchy.

Formally, we denote the set of concept types as T_C, the set of relation types as T_R and the set of individual markers that are used to labeled the concept nodes as $\mathrm{markers}$, which defines a vocabulary $\mathcal{V} = (T_C, T_R, \mathrm{markers})$. A basic conceptual graph G is then defined by a 4-uple $G = (C_G, R_G, E_G, l_G)$, where

- (C_G, R_G, E_G) is a finite undirected and bipartite multigraph. C_G is the set of concept nodes. R_G is the set of relation nodes, and E_G is the set of edges.
- l_G is a naming function of the nodes and edges of the graph G which satisfies:
  1) A concept node c is labeled with a pair $l_G(c) = (type(c), marker(c))$, where $type(c) \in T_C$ and $marker(c) \in \mathrm{markers} \cup \{\star\}$.
  2) A relation node r is labeled by $l_G(r) \in T_R$. $l_G(r)$ is also called the type of r.

C. Specialization and generalization of graphs

A specialization/generalization relationship is defined on the graphs. These relationships are used for the query function. The aim of the query is indeed to find all the sub graphs of the information graph that are specializations of the query graph. Therefore, the query is expressed as a generic graph.

We define these relationships hereafter.

1) Relationships between conceptual types: Given the hierarchical nature of the vocabulary, a partial order holds among the set of conceptual types T_C, interpreted as a relation of specialization: $t_1 \leq t_2$ means that t_1 is a specialization of t_2, that is to say that any instance of the class denoted by t_1 is also an instance of the class denoted by t_2.

2) Relationships between concepts: Given the order on T_C, we can also partially order the concepts that are defined on $T_C \times \{\mathrm{markers} \cup \{\star\}\}$, by a specialization relation as follows. Let $c_1 = [T_1 : m_1]$ and $c_2 = [T_2 : m_2]$ be two concept nodes, we define:

$$c_1 \leq c_2 \quad \text{iff} \quad \begin{cases} T_1 \leq T_2 \\ m_2 = \star \quad \text{or} \quad sim(m_1, m_2) \geq thres \end{cases} \qquad (1)$$

where sim is a similarity function and $thres$ a user-defined threshold.

For instance, if we consider that the conceptual type `Event` is greater (i.e. more general) than the conceptual type `TerroristEvent`, we have:

`[Event:*]` $\geq$ `[Event:e1]` $\geq$ `[TerroristEvent:e1]`
but `[Event:e1]` and `[TerroristEvent:*]` are not comparable.

3) Relationships between graphs: We also define a specialization relation between graphs. This relation is denoted by $\sqsubseteq$ (in order to avoid confusion with the specialization relation $\leq$ between concepts). Let A and B be two basic conceptual graphs. $\mathcal{C}_A$ and $\mathcal{R}_A$ denote the set of concepts and relations of the graph A, defined over the vocabulary $\mathcal{V}$. Denoting as P_{AB} the set of graph isomorphisms between A and B, we have:

$$A \sqsubseteq B \Leftrightarrow \exists p \in P_{AB}, \begin{cases} p : \mathcal{C}_A, \mathcal{R}_A & \rightarrow & \mathcal{C}_B, \mathcal{R}_B \\ c_A, r_A & \mapsto & c_B, r_B \\ \forall c_A \in \mathcal{C}_A, & & c_B \leq c_A \\ \forall r_A \in \mathcal{R}_A, & & r_B = r_A \end{cases}$$

By extension of the notation, $G_A|_{r_A} \sqsubseteq G_B|_{r_B}$ denotes that the subgraph of G_A restricted to the relation node r_A and its linked concept nodes is a generalization of the subgraph of G_B restricted to the relation node r_B and its linked concept nodes.

D. Graph based Fusion: Information Query

The InSyTo Synthesis platform encompasses a generic graph based fusion algorithm that is used for the three functions (information fusion, information synthesis and information query). The usage of the algorithm (parameters and launch mode) determines the function that is realized.

The fusion algorithm is made of two interrelated components (see Figure 3). The first component is a generic subgraph matching algorithm, which itself relies on the use of fusion strategies.

The graph matching component takes care of the overall structures of the initial and fused observations. It is in charge of the structural consistency of the fused information, regarding the structures of the initial observations, within the fusion process.

The fusion strategy part is made of similarity, compatibility and functions over elements of the graphs to be fused (see equation 1 for instance). They enable the customization of the generic fusion algorithm according to the context in which it is used. The context encompasses the application domain, the semantics of the information items and user preferences. The fusion strategies enable to manage the discrepancies that may be observed in observations of the same situation by different sources (see [6]).

Within the query function, the strategies that are used in the algorithm follow the specialization/generalization relationships defined before (II-C). We use a *subsomption* strategy and a whole-structure conservation mode.

The subsomption strategy is used within the nodes to nodes comparison between the query and the data graphs. That is to say that the concepts of the query graph must be more general than the ones of the data graph in order to be fused.

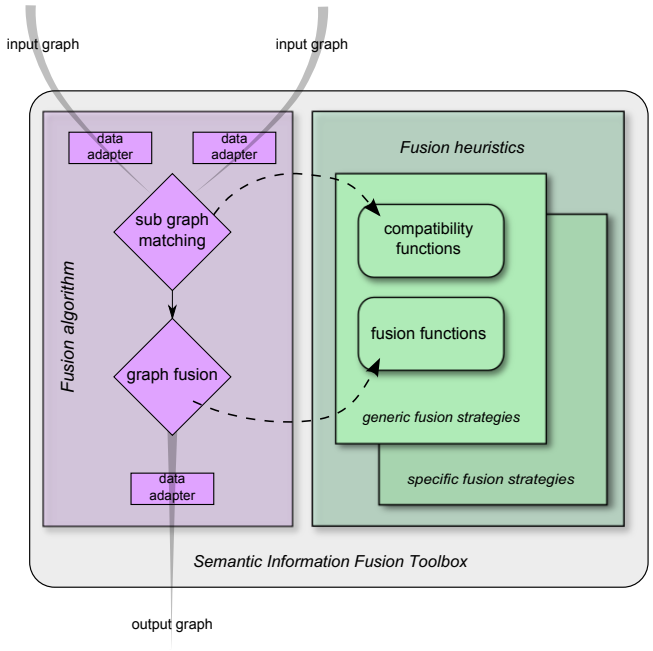

Fig. 3. InSyTo algorithm

The specialization relationship between the query and the data graphs also imply that the structure of the query graph must be entirely found in the data graph. In other words, all the relations of the query graph must have an image in the data graph, that respects the structure of the query graph. The query function relies on the search for injective homomorphism between the query graph and the data graph.

III. Towards bigger graphs

A. Hadoop : a framework for processing "big data"

Hadoop is a implementation of a distributed file system along with a model for performing distributed data processing on this file system.

Hadoop's file system is called HDFS (Hadoop Distributed File System). It allows the partitioning of data across many nodes. A Hadoop cluster scales computation capacity, storage capacity and I/O bandwidth by simply adding commodity servers.

The data processing model is called MapReduce, and allows processing of large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. The advantage of MapReduce is that the processing can be performed in parallel on multiple processing nodes (multiple servers) so it is a system that can scale very well.

Since it's based on a functional programming model, the map and reduce steps each do not have any side-effects (the state and results from each subsection of a map process does not depend on another), so the data set being mapped and reduced can each be separated over multiple processing nodes.

The use of Hadoop for the parallelization of our algorithms has two advantages. The first one is that the graphs are not loaded in memory, but sub-parts of them are passed along the processing nodes thanks to the distributed file system. The second one is that parts of the isomorphism search process itself can be distributed on the different processing nodes, enabling a speed-up of the overall process.

B. Problem decomposition

In this work, we focus on the query functionality of InSyTo Synthesis. The input data are two graphs. The *data graph* is a big network (more than 20,000,000 nodes). The *query graph* is a relatively small graph. Indeed, the query is "written" by the analyst and contains most of the time 20 to 50 nodes.

The main issue of the subgraph isomorphism algorithm is the combinatoric explosion and the big graph structures. The management of the structural part of the graphs (i.e. which node is linked to which one) is a difficult problem while looking for candidate images for the nodes of one graph in the other. Therefore, within the search for candidate images for the nodes of the query graph in the information graph, our approach is to split up the process into a first phase that will manage the values of the query and information nodes and a second phase in charge of the management of the structural aspects of the graphs.

In order to follow the MapReduce model, we cut off the two graphs (data and query) into small pieces that will then be pairwise compared. The comparison take into account the value of the nodes themselves, without taking care of the overall structures of the graphs. Therefore, the comparisons are independent from one another, and the MapReduce model can be applied.

Figure 4 shows the decomposition of a query graph into two small graphs. Each small graph corresponds to one of the relation nodes of the query graph, linked to a copy of its neighbor concept nodes.

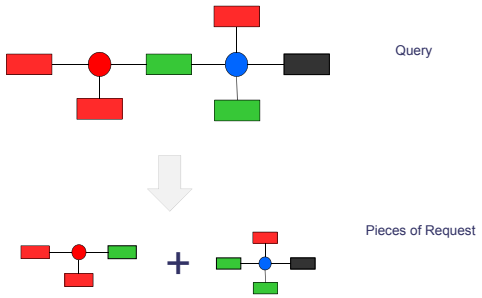

Fig. 4. Decomposition of a 'query' graph

Once the two graphs are cut around the relations nodes, each relation node and the ordered list of concepts nodes it is linked to, constitutes an entry for the MapReduce algorithm. To be readable by the Hadoop framework, the set of pieces of the query are written into their linear form (see figure 2) into a file. Each line of the file is one of the subgraph and will be processed by the MapReduce algorithm.

The MapReduce algorithm looks for all the candidate images for each piece of the query (from the query graph) in the set of pieces of data (from the data graph) (see Figure 5).

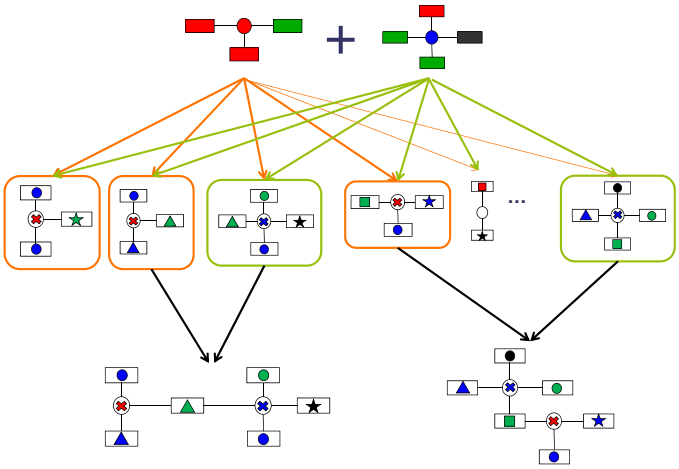

Fig. 5. Finding candidate images and reconstructing answer graphs

The master node of the Hadoop cluster divides the data graph into a set of relation nodes, each being linked a set of concepts. It then distributes the compatibility test of the data relations against the query graph to the worker nodes of the cluster. The worker nodes process the compatibility test and produce, if the test is achieved, output records R as pairs of compatible query and data nodes.

The map function on a query graph $G_r = (C_r, R_r, E_r, l_r)$ and a data graph $G_d = (C_d, R_d, E_d, l_d)$ is detailed hereafter. The compatibility between a subgraph G_{r,r_i} of the query G_r and a subgraph G_{d,d_i} of the data graph G_d follows the specialization relationship $\sqsubseteq$ defined on subgraphs restricted to one relation (see II-C).

Algorithm 1 Map function for a query graph and a data graph

function map(G_r, G_{d,d_i}) :
$R \leftarrow \emptyset$
for all $G_{r,r_i} \in G_r$ **do**
  if $G_{r,r_i} \sqsubseteq G_{d,d_i}$ **then**
    {append output record}
    $R \leftarrow R \| (G_{r,r_i}, G_{d,d_i})$
  end if
end for
return R

Once the map step is achieved, the reduce step takes place. The master node of the processing cluster collects the records $R = \{(G_{r,r_i}, G_{d,d_i}) \forall G_{r,r_i} \in G_r | G_{r,r_i} \sqsubseteq G_{d,d_i}\}$ produced during the map phase and combine them so as to associate each subgraph G_{r,r_i} of the query with the list of all compatible subgraphs G_{d,d_i} of the data graph.

The reduce function aggregates the set of candidate subgraphs G_{d,d_j} as a list of images of G_{r,r_i}. The produced associative array $r\_img(G_{r,r_i}) = \{G_{d,d_j} \forall G_{d,d_j} \in G_d | G_{r,r_i} \sqsubseteq G_{d,d_j}\}$ has the same set properties than R but permits to improve further query times.

C. Result graphs build-up

Once the candidate answers to each piece of query are processed, we need to build up answer graphs from these

Algorithm 2 Reduce function on the set of candidate subgraphs

function reduce(R) :
$r\_img \leftarrow \emptyset$
for all $(G_{r,r_i}, G_{d,d_i}) \in R$ **do**
  {append subgraph to the list mapped to the query graph}
  $r\_img(G_{r,r_i}) \leftarrow r\_img(G_{r,r_i}) \| G_{d,d_i}$
end for
return r_img

candidates. The different combinations of pieces of answers are provided, taking care of the original structure of the data graph (see Figure 5). The candidate pieces for answers are assemble one with another, if and only if their association respects the structural constraints of the initial data graph. In other words, the connectivity between the relations through the concept nodes is checked while building the answer graphs.

Within the result graph reconstruction algorithm, the following notations are used:

- query graph $G_r = (C_r, R_r, E_r, l_r)$;
- data graph $G_d = (C_d, R_d, E_d, l_d)$;
- $fstrategy$ is the fusion strategy that is used for compatibility conditions and fusion functions. The fusion strategy used within the query function is the subsomption of concept nodes.
- $r\_img = \{(G_{r,r_i}, G_{d,r_j}) | G_{r,r_i} \sqsubseteq G_{d,r_j}\}$ is the associative array containning the records generated by the MapReduce algorithm.
- G_{r,r_i} is the subgraph of G_r restricted to the relation r_i and its linked concept nodes $c_j \in C_r$;
- $r\_mods = \{G_{r,r_i} | r_i \in R_r\}$ is the set of subgraphs of G_r restricted to each relation r_i of G_r.
- $ans = \{G_{ans_i} | G_{ans_i} = subgraph(G_d) \wedge G_r \sqsubseteq G_{ans_i}\}$ the set of subgraphs of G_d that are answers to the query (i.e. more specific than) G_r.
- $asso_i = \{(G_{r,r_i}, img_i)\}$ is an associative array that associates each subgraph restricted to a relation of the query graph G_r to a subgraph restricted to a compatible relation of the data graph G_d that is its image in the current association. An image of a relation node can be added in an association if and only if it is compatible (regarding node values and graph structures) with the already associated relations.
- $assos = \{asso_i\}$ is the set of all association tables $asso_i$ containing relations nodes of the data graph G_d associated to the relation nodes of query graph G_r and that respect the conditions given by G_r and the fusion strategy $fstrategy$
- $graph(asso_i) \rightarrow G_{answer_i}$ is a function that transforms an association table $asso_i$ into a graph by reconstructing the graph structure.

The input of the algorithm are r_img, the table of potential relation images of the data graph for each relation node of the query and the fusion strategy $fstrategy$ that relies on

the subsomption between concept nodes and graphs (see II-C).

Algorithm 3 Result graph reconstruction algorithm

associateModel($r\_img, fstrategy$) :
$assos \leftarrow \emptyset$
for all $img \in r\_img(r\_mods_0)$ **do**
  {append an element to the associations:}
  $assos \leftarrow assos \| (r\_mod_0, img)$
end for
for $r\_mod \in \{r\_mods_1 \dots r\_mods_n\}$ **do**
  $i \leftarrow 0$
  while $i + 1 < |assos|$ **do**
    {iterate over $assos$ while modifying it}
    $i \leftarrow i + 1$
    $found \leftarrow \bot$
    for all $img \in r\_img(r\_mod)$ **do**
      if $compatible(assos_i, (r\_mod, img, fstrategy))$ **then**
        $associate(assos_i, (r\_mod, img, fstrategy))$
        $found \leftarrow \top$
      end if
    end for
    if $\neg found$ **then**
      {remove the i[th] association:}
      $assos \leftarrow \{assos_0, \dots, \widehat{assos_i}, \dots, assos_n\}$
    end if
  end while
end for
$ans \leftarrow \emptyset$
for all $asso \in assos$ **do**
  $ans \leftarrow ans \| graph(asso)$
end for
return ans

IV. Experimentations

A. The Global Terrorism Database

The global terrorism database[1](GTD) is an open-source database that contains the descriptions of all the terrorist events that occurs worldwide between 1970 and 2012.

The GTD has the following characteristics :

- It contains information on over 113,000 terrorist attacks;
- It includes information on more than 47,000 bombings, 14,400 assassinations, and 5,600 kidnappings since 1970;
- It includes information on 45 to more than 120 variables for each case;

The GTD is supervised by an advisory panel of 12 terrorism research experts and over 4,000,000 news articles and 25,000 news sources were reviewed to collect incident data

For each GTD incident, information is available on the date and location (region, country, state, city) of the incident, perpetrator group name, tactic used in attack, the weapons used and nature of the target, the number of casualties, etc.

Other variables provide information unique to specific types of cases, including kidnappings, hostage incidents, and hijackings.

This database represents a huge work and is particularly valuable for intelligence analysts. Therefore smart functionalities that would enable the recognition of modus operandi, for instance, are eagerly awaited.

Within our experimentations, we imported the data contained in the GTD into an information network. The importation of the data is agnostic to the contents of the database.

The GTD database is a classical "table". Each line of the table contains the description of an event, each column of the table corresponds to a characteristic of the events (date, location, attack type, etc.). To import the GTD data into the conceptual graphs format, we first automatically built the type hierarchy described in II-B. Therefore, Two concept types were defined, `GTDEvent` and `GTDEntity` that are directly attached to the root type of the hierarchy. The types of relations are extracted from the column labels of the table.

The data itself is then imported into a graph structure by creating a `GTDEvent` which value is the identifier of the event contained in the first column of the table. This concept is then linked to concepts of type `GTDEntity` and value to values of the other columns through relation nodes of the types corresponding to the labels of the columns.

The resulting network of information contains more than 10,000,000 concept nodes linked through 13,560,000 relation nodes.

B. Results

We launched several queries on the network made of the GTD data. These queries are patterns of terrorists events containing some known and unknown values. The figure 6 depicts an example of such a query.

It queries the network for information on years, types of weapons used and number of persons killed in all the terrorist events perpetrated by Tamils and that occurred in Sri Lanka. This query gave us 2935 answers under the form of graphs describing these terrorists events (see figure 6 for one of the answers).

In order to test the scalability of our implementation, we artificially increased the size of the information network by generating 7 terrorist events for each event described in the GTD. This generated a text input file of 6.64 Mo that describes the 1,560,000 events. The requests were sent on a single core CentOS 5.7 machine with 3Gb RAM and 20Gb disk memory. The answers to requests were obtained in less than 1 minute.

The results obtained during this first experimentation phase are promising. The whole data network can be processed through this new parallelized version of the fusion algorithm. This was not worth considering, using a classical algorithm for the maximum subgraph isomorphism problem.

The structure of the hierarchy that was built from the GTD database is flat. Only two types of concepts are considered,

[1]The National Consortium for the Study of Terrorism and Responses to Terrorism (START) makes the GTD available via an online interface on the website of the project : http://www.start.umd.edu/gtd/

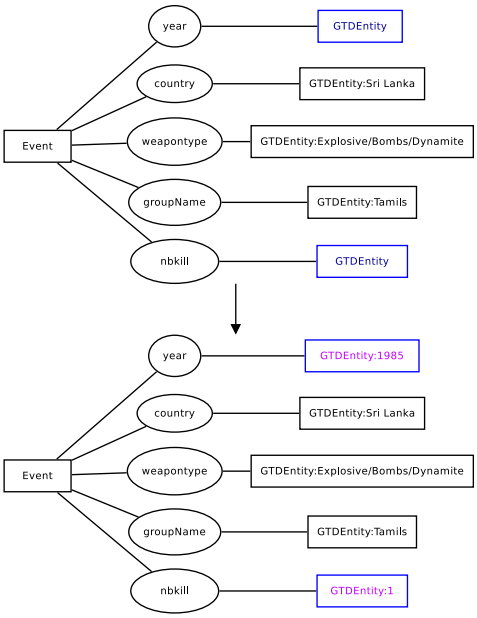

Fig. 6. Example of query and answer on the information network

with no hierarchical dependency between them. The advantage of the agnostic importation of the data is that the experimentation platform can be used on any other database. However, it also prevents from taking advantages of all the aspects of the conceptual graphs based information fusion approach. The queries made on the network could be done efficiently on the table version of the database, as the structure was adapted to the contents of the data. The advantages of using the hierarchy of concept types and fusion strategies in order to query complex inexact patterns on the network where shown in previous work ([22] and [8]) however.

V. Conclusion and Future work

In this paper we present the implementation of a distributed algorithm for the graph based information fusion. The conceptual graphs based information fusion platform that was previously developed enables the management of inconsistencies withing information fusion. To do that, the platform uses a hierarchy of the types of nodes of the data network, as well as fusion strategies. The fusion strategies encompass similarity measures and fusion functions that enable managing the discrepancies between values of the information nodes of a network of information.

The new algorithm relies on the Map/Reduce principle, which enables the distribution of part of the process on several processing nodes. The implementation uses the Hadoop framework.

Tests were conducted on an information network made of the data contained in the Global Terrorist Database. The tests consisted in querying specific patterns of terrorist events. The information network contains more than 20,000,000 nodes linked together. Using Hadoop first enables to distribute the process among several processing nodes. It secondly avoid loading the whole information network in memory, by splitting and distributing on the Hadoop file system.

The second phase of the algorithm is still time consuming, which underlines the necessity of feeding it with as few inputs as possible and consider to distribute it on several processing nodes. Therefore, these results lead us two the two following trail for future work:

- How to optimize the research for potential images for each subgraph of the query restricted to the relation nodes ?
- How to parallelize the result graphs build-up ?

The present paper describes the implementation of the query functionality of the information synthesis platform. Future work will encompass the implementation of the other functions of InSyTo, namely information fusion and information synthesis. This should follow the same fusion process.

Acknowledgment

The research leading to these results has received funding from the European Union Seventh Framework Program (FP7/2007-2013) iSAR+ FP7-SEC-2012-1-312850

References

[1] J. Llinas, R. Nagi, D. Hall, and J. Lavery, "A multi-disciplinary university research initiative in hard and soft information fusion: Overview, research strategies and initial results," in *Information Fusion, 13th International Conference on*, 2010.

[2] G. Gross, R. Nagi, K. Sambhoos, D. Schlegel, S. Shapiro, and G. Tauer, "Towards hard+soft data fusion: Processing architecture and implementation for the joint fusion and analysis of hard and soft intelligence data," in *Information Fusion, 15th International Conference on*, 2012.

[3] K. Date, G. A. Gross, S. S. Khopkar, R. Nagi, and K. Sambhoos, "Data association and graph analytical processing of hard and soft intelligence data." in *Proceedings of the 16th International Conference on Information Fusion, FUSION 2013, Istanbul, Turkey, July 9-12*. IEEE, 2013, pp. 404–411. [Online]. Available: http://dblp.uni-trier.de/db/conf/fusion/fusion2013.html#DateGKNS13

[4] N. Belov and P. Gerken, "Mixed initiative soft data fusion associate," in *Information Fusion, 12th International Conference on*, 2009.

[5] H. Kohler, D. A. Lambert, J. Richter, G. Burgess, and T. Cawley, "Implementing soft fusion." in *Proceedings of the 16th International Conference on Information Fusion, FUSION 2013, Istanbul, Turkey, July 9-12*. IEEE, 2013, pp. 389–396. [Online]. Available: http://dblp.uni-trier.de/db/conf/fusion/fusion2013.html#KohlerLRBC13

[6] C. Laudy, *Semantic Knowledge Representations for Soft Data Fusion — Efficient Decision Support Systems - Practice and Challenges From Current to Future*. Chiang Jao Publisher, 2011.

[7] S. Fossier, C. Laudy, and F. Pichon, "Managing uncertainty in conceptual graph-based soft information fusion," in *Proceedings of the 16th International Conference on Information Fusion, FUSION 2013, Istanbul, Turkey, July 9-12*, 2013, pp. 930–937.

[8] C. Laudy, É. Deparis, G. Lortal, and J. Mattioli, "Multi-granular fusion for social data analysis for a decision and intelligence application," in *Proceedings of the 16th International Conference on Information Fusion, FUSION 2013, Istanbul, Turkey, July 9-12*, 2013, pp. 1849–1855.

[9] B. D. McKay and A. Piperno, "Practical graph isomorphism, II," *Journal of Symbolic Computation*, vol. 60, pp. 94–112, Jan. 2014. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0747717113001193

[10] T. Plantenga, "Inexact subgraph isomorphism in MapReduce," *Journal of Parallel and Distributed Computing*, vol. 73, no. 2, pp. 164–175, Feb. 2013. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0743731512002559

[11] Z. Zhao, G. Wang, A. R. Butt, M. Khan, V. A. Kumar, and M. V. Marathe, "SAHAD: subgraph analysis in massive networks using hadoop." IEEE, May 2012, pp. 390–401. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6267876

[12] Y. Liu, X. Jiang, H. Chen, J. Ma, and X. Zhang, "MapReduce-Based pattern finding algorithm applied in motif detection for prescription compatibility network," in *Advanced Parallel Processing Technologies*, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, Y. Dou, R. Gruber, and J. M. Joller, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, vol. 5737, pp. 341–355. [Online]. Available: http://link.springer.com/10.1007/978-3-642-03644-6_27

[13] J. R. Ullmann, "An algorithm for subgraph isomorphism," *J. ACM*, vol. 23, no. 1, pp. 31–42, Jan. 1976. [Online]. Available: http://doi.acm.org/10.1145/321921.321925

[14] D. Conte, P. Foggia, and M. Vento, "Challenging complexity of maximum common subgraph detection algorithms: A performance analysis of three algorithms on a wide database of graphs," *J. Graph Algorithms Appl.*, vol. 11, no. 1, pp. 99–143, 2007.

[15] H. Bunke, P. Foggia, C. Guidobaldi, C. Sansone, and M. Vento, "A comparison of algorithms for maximum common subgraph on randomly connected graphs," in *Lecture Notes on Computer Science*. Heidelberg: Springer-Verlag, 2002, vol. 2396, pp. 123–132.

[16] J. Raymond and P. Willett, "Maximum common subgraph isomorphism algorithms for the matching of chemical structures," *Journal of Computer-Aided Molecular Design*, vol. 16, no. 7, pp. 521–533, 2002. [Online]. Available: http://dx.doi.org/10.1023/A%3A1021271615909

[17] S. N. Ndiaye and C. Solnon, "CP Models for Maximum Common Subgraph Problems," in *17th International Conference on Principles and Practice of Constraint Programming (CP)*, ser. LNCS. Springer, Sep. 2011, pp. 637–644. [Online]. Available: http://liris.cnrs.fr/publis/?id=5128

[18] C. Solnon, "Alldifferent-based filtering for subgraph isomorphism," *Artif. Intell.*, vol. 174, no. 12-13, pp. 850–864, Aug. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.artint.2010.05.002

[19] T. White, *Hadoop: The Definitive Guide*, 1st ed. O'Reilly Media, Inc., 2009.

[20] J. F. Sowa, *Conceptual Structures: Information Processing in Mind and Machine*. Reading: Addison-Wesley, 1984.

[21] M. Chein and M.-L. Mugnier, *Graph-based Knowledge Representation: Computational Foundations of Conceptual Graphs*. Springer, 2008.

[22] C. Laudy, "Introducing semantic knowledge in high level information fusion," Ph.D. dissertation, Paris 6 University, 2010.