Computational requirement of schema matching algorithms

PETER MARTINEK, BELA SZIKORA
Department of Electronics Technology
Budapest University of Technology and Economics
Goldman Gy. tér 3., 1111 Budapest
HUNGARY
martinek@ett.bme.hu

Abstract: - The integration of different data structures e.g. relational databases of information systems is a current issue in the area of information sciences. Numerous solutions aroused recently aiming to achieve a high accuracy in similarity measurement and integration of schema entities coming from different schemas. Researches usually properly evaluate the capabilities of these approaches from the point of view of accuracy. However the computational complexity of the proposed algorithms is hardly ever examined in the most of these works. We claim that efficiency of a solution can only be judged by taking into account both the accuracy and the computational requirements of participating algorithms. Since there are many known measurement methods and metrics for the evaluation of accuracy, the focus is set for the analysis of their computational complexity in this paper. After the problem formulation the main ideas behind our method are presented. Various approximation techniques and methods of applied algorithm theory are used to evaluate the different approaches. Three specific approaches were also selected to present the work of our method in details on them. Experiments run on several test inputs are also included.

Key-Words: - Computational complexity, Schema matching, Approximation techniques in computational requirement estimation

1 Introduction

Integration of different software systems and corresponding data structures is a certain need in information technology. The field of its applications is various starting from medical systems to enterprise application integration through E-commerce and E-government. Because system integration is hardly separable from the mapping of different data structures, the schema matching problem i.e. comparing and aligning of entities of schemas to each other is a current issue as well.

The aim of schema matching algorithms is to identify the semantic relevance between the entities of different data representation structures coming from different systems. Most of the proposals contain a similarity measure component, which returns the connection strength between two objects in the form of a number between 0 and 1. Pairs of entities having a value near to 1 probably represent the same real world concepts. Hence they must be connected during the schema or application integration task. On the other hand, entities described by lower values are out of interest from the point of view of integration. Similarity measurement methods can save a lot of human efforts this way. Thus preventing the necessity of the presence of a human expert during the schema matching task is also a reasonable goal by developing different approaches and methodologies.

There are 3 basic types of schema matching approaches today:

- Linguistic approaches examine the naming similarity of entities using different string comparing functions for example searching for sub-strings or concatenations. Usually they are also extended by (domain specific) dictionaries and taxonomies to be able to detect the similarities in the meaning of schema concepts as well.
- Structural methods are based on the comparing of paths connected to the given entities leading to the leaves, children or to the root element. The main idea behind this approach is, that two entities of two different schemas probably represent the same real world entity if their structural neighborhood is built similarly e.g. the two paths leading to the root element are similar. Path similarities are mainly measured by defining indicators for similar node correspondence, node order, etc.
- Combining the two approaches above and applying more specific algorithms within solid frameworks results in a solution called combined approach. Because of its robustness and effectiveness, most of the presented solutions are from this category in current literature. However algorithm complexity and computational costs are hardly taken into account.

There are many possible area of application e.g. aligning of service interfaces in a SOA based integration scenario, where schema matching algorithms should be performed quite often during the everyday work. In this case, it is important to have an algorithm, which is able to be executed in an acceptable time. Therefore analyzes

and prediction of algorithms' runtime is at least as essential as the evaluation of the accuracy of their results.

This paper presents mathematical methods and techniques to predict computational costs and hence execution run-time of schema matching approaches. The structure of the paper is as follows: the next chapter summarizes works related to ours. In chapter 3 we present the problem of computational requirement estimation and describe our solution. Chapter 4 contains experimental results validating our method. Chapter 5 briefly describes algorithm accuracy and finally chapter 6 concludes the achieved results.

2 Related work

There are numerous researches in literature about comparison and evaluation of different data sources and their schemas [1, 5, 8, 11, 15, 16, 17,18, 19].

An automated schema matching solution working on XML schemas is presented in [14]. The described method combines linguistic and structural similarity extended with the evaluation of data type compatibilities. Within the linguistic part abbreviations and acronyms are also identified while prepositions and articles are disregarded due to domain-specific dictionaries constructed for the examined schemas.

A special approach, called similarity flooding is presented in the work [12], which is hard to be identified upon the classification of schema matching approaches introduced above. The main idea behind the similarity flooding is that the similarity of two given entities can be measured by the similarity of their neighborhoods. The paper also presents detailed analyzes and comparison of results to other approaches. However this is performed only from the point of view of accuracy and the paper lacks any kind of estimation of run-time costs. This algorithm is described in details and compared with our approach later in chapters 3 and 4.

The authors in [4] present a conventional example of a combined approach. The evaluation starts with a linguistic analyzes based on the open dictionary called WordNet[6], but the main added value is the comprehensive structural method performed on the schema trees. Unfortunately this paper also lacks of computational complexity estimation.

Although this requires the handling of functions working also on graphs, matrices and other ordered data structures, our solution is able to predict the computational requirement also for this approach – see later in chapters 3 and 4.

The presented approach called Cupid in [9] also uses a wider set of techniques including discovering mappings based on their names, data types, constraints and schema structure. The authors constitute that most of the useful information can be found in the leaves of schemas. Thus the similarity of the leaf context is highly weighted in the calculations. The number of one to one comparisons is decreased by a separately clustering of concepts into categories at an early stage of the method. Otherwise the different implementations e.g. the structural similarity evaluation of this approach are quite similar to the previous one preventing me from the further description and comparison of them with other approaches. Furthermore we can also judge approaches against two other proposals (DIKE[14] and MOMIS[2]) indirectly, because Cupid definitely outperforms both of them.

In [13] the authors present a schema matching method working on XML schemas. Similarly to the approach Cupid the evaluation starts with the clustering of schemas into various groups. The syntactic similarity measurement is performed in 3 steps namely preprocessing, data mining and postprocessing while a specific graph representation called dendogram facilitates the generalization and specialization processes of the clusters to develop an appropriate schema class hierarchy. Unfortunately the analysis of the results is restricted to the parameterization of the presented approach and is only presented in the unique metrics of the paper. This hinders the comparison with other approaches including mine. However taking into account the size of evaluated schemas and the values of applied efficiency indicators the performance should be at the same level as the methods in [12] and [9].

The authors of [3] propose an algorithm based on tree similarity matching methods in a multi agent environment for a classical buyer-seller scenario. The similarity algorithm itself is implemented as a recursive functional program in a language called Relfun. Although this may save some computational costs in the run-time, an additional transformation step of classical XML based inputs i.e. the XML serialization in Object-Oriented RuleML is needed. Because there are no experiments on schemas of realistic sizes presented, the exact comparison of this approach with others would require enormous amount of work. Taking into account the applied structural algorithm and comparing it with other approaches the performance of this approach can be similar (or maybe slightly better in run-time) to the methods in [12] and [9].

A generic schema matching tool called COMA++ is presented in [7]. It provides a library of individual matchers realizing a flexible platform for a combined matcher. The application of different matching strategies and the decomposition of large schemas with a fragment-based matcher into smaller sets ensure high scalability for the presented schema matching solution. The approach is also evaluated on schemas from various sizes (containing a number of nodes between 27 and 843

and a number of possible paths between 34 and 26228) and it is also compared with other proposals extensively. However the focus is mainly set on accuracy comparison while it a proper and detailed analyzes of run-time cost is still missing.

3 Algorithm complexity of approaches

Besides of the expected accuracy of the results the run time cost of an algorithm is also a key aspect by finding a solution for the schema matching problem. More accurate solutions may require much more resources (e.g. computational performance) to provide results. This can lead to enormous long run-time in the given hardware configurations which can not be accepted by some system e.g. online, real time systems or even by a solid development environment.

The costs of an algorithm are strictly connected with the complexity of the given method. We have proved that complexity is proportional by the expected number of the steps at the execution. Because there is a (single) computer operation in the background of each performed step, the actual run-time of given computer configurations can also be predicted for given schemas this way. Hence the number of the expected steps is calculated by our method. In the next few paragraphs we present the main tasks and tools of our method. For better understanding tasks of real schema matching solutions are also described and evaluated.

3.1 Examples for schema matching approaches

The proposal *similarity flooding* is based on the following idea: if a graph is constructed where nodes represent entities like complex types or attributes and edges represent relationship among them like containing, inheritance or association, the relation between two given entity of the schemas is determined by the relation of their neighborhood. In other worlds, if the nodes close to entity A in the fist schema are semantically related to corresponding nodes close to entity B in the second schema, then entity A and entity B are probably semantically related and vice versa [4].

The key point of the approach is the algorithm for similarity flooding, where similarity of the nodes is distributed among the neighbors. This is performed for all nodes in a specially constructed pair wise connectivity graph (PCG). Furthermore this is an iterative step, which is stopped by reaching a stable state or a given number of ran iterative steps. The results (similarity values of entities creating common nodes in the PCG) are directly readable after flooding is stopped.

The next presented approach - called *WordNet* in our article- has a strong structural analyzer part but relies on the values of a dictionary based matcher, the WordNet [4, 6]. After the initial values of similarity between entities is evaluated based on the WordNet, a complex structural matching algorithm determines the relations between the entities reconstructed in directed acyclic graphs (DAGs).

We have also developed a schema matching solution in our former work [10]. The approach determines similarity values on three partial similarity functions. The name, connected term and attribute context similarities give the returning values by a weighted sum. The algorithm was designed to also perform well in everyday problem solutions. Hence its computational complexity is intended to be much lower than using other approaches. On the other hand its accuracy is also not lower so this approach – called *NTA* in the rest of the paper – definitely outperforms other presented algorithms, see later.

Every main steps of every approach will be presented and analyzed from the point of computational complexity in the next few sub-chapters.

3.2 Complexity of simple comparisons and constructional tasks

This section presents the computational complexity of some simpler tasks upcoming for example in the approach NTA.

By evaluating the linguistic correspondence between the naming of different complex types of schemas we use function $N(C_i,G_j)$, which is a single one-step operation – function substring. So the number of steps of function $N(C_i,G_j)$ is 1 for a given evaluation of two different complex types.

Calculating the similarity based on attached terms uses the function $T(C_i,G_j)$. This compares all attached concepts of evaluated couples what results $\|\mathrm{Term}(C_i)\| * \|\mathrm{Term}(G_j)\|$ number of steps, where $\|\mathrm{Terms}(C_i)\|$ is the number of terms connected to complex type C_i.

The correspondence between the attributes of complex types is calculated by the function $A(C_i,G_j)$. Within this every attribute of the local schema is compared with every attribute of the global schema resulting number of steps $\|\mathrm{Attr}(C_i)\| * \|\mathrm{Attr}(G_j)\|$ where $\|\mathrm{Attr}(C_i)\|$ represents the number of attributes connected to complex type C_i.

By the comparisons of two whole schemas all functions depicted above should be calculated to all pairs of the complex types of the schemas. This means

$$\text{Steps}_{\text{NTA\_Perform}}(C,G) = \\ = \sum_{i,j} 1 + \|\text{Term}(C_i)\| * \|\text{Term}(G_j)\| + \|\text{Attr}(C_i)\| * \|\text{Attr}(G_j)\| \quad (1)$$

steps for schemas C and G.

These expressions should be applied for every simple comparing steps of every schema matching approaches e.g. the retrieving of the similarity values from the dictionary in proposal Wordnet or by the creation of initial values between the OIM graph nodes in the similarity flooding.

Most of the algorithm requires some preparation e.g. constructing trees from complex types before its execution. This requires the processing of all complex types with their all attributes which costs

$$\text{Steps}_{\text{NTA\_Prepair}}(C,G) = \sum_i \|\text{Attr}(C_i)\| + \sum_j \|\text{Attr}(G_j)\| + \|C\| + \|G\| + \\ + \|C\| * \|\text{Term}(C_i)\| + \|G\| * \|\text{Term}(G_j)\| \quad (2)$$

where $\|C\|$ is the number of the complex types in the first schema and $\|G\|$ denotes the number of complex types in the second schema. So similar to the computational requirements of preparation in approach NTA, every preparation steps of every matching algorithms should be estimated with the expression above.

By the similarity flooding an open information model (OIM) based graph is constructed both from the local schemas. Similarly to initial graph construction by the NTA approach this step requires

$$\text{Steps}_{\text{OIM}}(C,G) = 2 * \left(\sum_i \|\text{Attr}(C_i)\| + \sum_j \|\text{Attr}(G_j)\| + \right. \\ \left. + \|C\| + \|G\| + Y_C + Y_G + 3 \right), \quad (3)$$

where Y_c (Y_G) is the number of existing data types in the first schema (second schema respectively) and the constant value of 3 is the insertion cost of the tree special category node (complex type, attribute, and attribute type) at the end of the expression.

The computational complexity of tree construction step of the approach Wordnet can also be predicted this way. So the first step of approach Wordnet requires

$$\text{Steps}_{\text{TreeConstr}}(C,G) = \sum_i \|\text{Attr}(C_i)\| + \sum_j \|\text{Attr}(G_j)\| + \|C\| + \|G\| \quad (4)$$

computational steps.

The second task of approach Wordnet should be mentioned here as well. Semantic relevance of entities of both schemas is retrieved from the Wordnet database in a number of steps as follows:

$$\text{Steps}_{\text{WordNET}}(C,G) = \left(\|C\| + \sum_i \|\text{Simple\_Attr}(C_i)\| \right) * \\ \left(\|G\| + \sum_j \|\text{Simple\_Attr}(G_j)\| \right) \quad (5)$$

3.3 Complexity of special graph traversals

An initial mapping between the nodes of the two OIM graphs is calculated in approach similarity flooding. This will represent a rough starting value for the similarity of nodes (used later by the initializing of the flooding algorithm at the PCG). The generated number of nodes to each complex type is 2 (1 for the entity and 1 for its name). Similarly, all attributes and possible range types will also be presented by 2 nodes in the OIM. Hence the number of generated nodes is 2 times the number of complex types plus 2 times the number of attributes plus 2 times the number of data types existing in the given schema.

Besides, there are exactly 2 nodes generated to every complex type and 3 initial nodes for signing complex type, attribute and data type category nodes. During the creation of initial mapping every node of the 2 generated OIM schemas must be compared to every node of the other schema. Thus the number of the steps (comparisons) performing in this section is the following:

$$\text{Steps}_{\text{Init\_map}}(C,G) = \left(3 + 2\|C\| + 2\sum_i \|\text{Attr}(C_i)\| + 2Y_c \right) * \\ * \left(3 + 2\|G\| + 2\sum_j \|\text{Attr}(G_j)\| + 2Y_G \right) \quad (6)$$

So the required steps of comparisons were calculated by determining the nodes created in each steps of the graph construction. This kind of deductions can also be used by analyzing of various matchers containing special graph-based structures.

Another typical example for this could be the evaluation of the PCG graph creation step by the similarity flooding approach. The pair wise connectivity graph (PCG) is created along the same type of edges of the two schemas. (The type of edges can be type, name, and other relation like attribute and data type). To do so, every edge of the same type from the two schemas is evaluated. Without presenting the detailed deduction here, the number of expected execution steps of the PCG creation is calculated by expression (7) as follows:

$$\text{Steps}_{\text{PCG}}(C,G) = 2\left(\|C\| + \sum_i \|\text{Attr}(C_i)\| + Y_C \right) * \left(\|G\| + \sum_j \|\text{Attr}(G_j)\| + Y_G \right) + \\ + \sum_i \|\text{Attr}(C_i)\| * \sum_i \|\text{Attr}(G_j)\| + \sum_i \|\text{Simple\_Attr}(C_i)\| * \sum_j \|\text{Simple\_Attr}(G_j)\|$$

To the estimation of the steps of the flooding itself we need to know the exact number of the nodes in the PCG graph. A deduction similarly to the above presented cases leads to expression (8):

$$\mathrm{Nodes}_{\mathrm{PCG}}(C,G) = 2 * \left(\|C\| + \sum_i \|\mathrm{Attr}(C_i)\| + Y_C \right) * \left(\|C\| + \sum_i \|\mathrm{Attr}(C_i)\| + Y_C \right)$$

Thus the flooding itself is performed on each node of the PCG the required number of its computational steps is as follows:

$$\mathrm{Steps}_{\mathrm{Flooding}}(C,G) = \sum_{\mathrm{steps}} \mathrm{Nodes}_{\mathrm{PCG}}(C,G) = \quad (9)$$

$$= \sum_{\mathrm{steps}} \left(2 * \left(\|C\| + \sum_i \|\mathrm{Attr}(C_i)\| + Y_C \right) * \left(\|C\| + \sum_i \|\mathrm{Attr}(C_i)\| + Y_C \right) \right)$$

where $\sum_{\mathrm{steps}}(...)$ signs that this must be performed for all iterative steps.

3.4 Complexity of structure comparisons

The structural similarity of graph nodes is calculated by 3 aspects by the approach Wordnet. The ancestor, child and leaf context all contribute to the final value of the similarity.

The ancestor similarity is calculated as follows:

$$\mathrm{Sim}_{\mathrm{ancest}}(C_i, G_j) = \mathrm{PS}(P_1, P_2) * \mathrm{LS}(C_i, G_j), \quad (10)$$

where $\mathrm{PS}(P_1, P_2)$ is the path similarity function for the path starting from entities C_i (G_j respectively) to the root element of the given trees and function $\mathrm{LS}(C_i, G_j)$ represents the result of WordNet request for C_i and G_j. Because the values of LS are already available only the two paths should be found. Because the computational complexity of the functions contained by the path similarity are in order $O(n^2)$, the a number of steps in this task is estimated as follows (11):

$$\mathrm{Steps}_{\mathrm{ancest}}(C_i, G_j) = \mathrm{length}(P_1) * \mathrm{length}(P_2),$$

Where function *length()* returns the number of nodes in a given path.

For the overall method of the calculation of ancestor context this must be performed for all possible pairs of complex types of the two schemas. However we face a serious problem by evaluating the expression above for real schemas: the length of the path is different value for every node of the tree. To be able to solve the problem, the length of paths within a schema is estimated by a constant value. For this we have chosen the average lengths to the root calculated as follows:

$$\mathrm{AVG\_length}_{\mathrm{root}}(C_i) = \sum_i \mathrm{length}_{\mathrm{root}}(P_i) / \|C_i\| \cdot \quad (12)$$

Using this value within the expression above the overall computational cost of the ancestor context calculation can already be predicted.

Based on our experiments we claim that this approximation method be a good estimation for all kind of algorithms where path analyzing (comparing) functions are used.

For example it is also applicable by the calculation of the leaf similarity. Due to the specification of the Wordnet approach, the path similarity (PS) must be calculated to all paths leading to (descendants) leaves of the given pair of concepts originating from the 2 different schemas. The computational cost of the evaluation of two entities is as follows:

$$\mathrm{Steps}_{\mathrm{leaves\_paths}}(C_i, G_j) = \sum_{\|\mathrm{leaf}(C_i)\| \|\mathrm{leaf}(G_j)\|} \mathrm{length}(P_1) * \mathrm{length}(P_2), \quad (13)$$

where similarly to the ancestor calculation the length is estimated with an average value of paths to the leaves.

The child context is also based on the function LS. However it requires no path similarity functions because only the set of associated children nodes are compared in this step.

3.5 Computational complexity of sorting lists

There are many situations where a simple sorting of a list of values is required by a task of a schema matching algorithm. The algorithm theory has all the necessary results which should be adopted here. However we should not forget about the exact type of applied ordering method. In our implementation we used a kind of quick-sort algorithm whose computational cost can be estimated by the following average value:

$$\mathrm{Steps}_{\mathrm{quick\_sort}} = n * 1{,}39 * \log_2 n \quad (14)$$

By the calculation of the exact value of child and leaf similarities only the highest 25% (or 50%) of the above presented functions must be taken into account. This requires the sorting of the results, which is

$$\mathrm{Steps}_{\mathrm{sorting\_child}}(C_i, G_j) \approx \left(\|\mathrm{child}(C_i)\| * \|\mathrm{child}(G_j)\| \right) * \\ * 1{,}39 \log_2 \left(\|\mathrm{child}(C_i)\| * \|\mathrm{child}(G_j)\| \right) \quad (15)$$

for the child context and

$$\mathrm{Steps}_{\mathrm{leaves}}(C_i, G_j) \approx \|\mathrm{leaf}(C_i)\| \|\mathrm{leaf}(G_j)\| * \\ * 1{,}39 \log_2 \|\mathrm{leaf}(C_i)\| \|\mathrm{leaf}(G_j)\| \quad (16)$$

for the leaf context.

3.6 Number of expected steps of the approaches

In this paragraph we present the expected number of steps of each analyzed approach in a complex form without detailed explanation. Deductions and detailed expressions can be constructed based on the specification of the approaches and the expressions and methods presented above.

$$\text{Steps}_{\text{NTA}}(C,G) = \text{Steps}_{\text{NTA\_Perform}}(C,G) + \text{Steps}_{\text{NTA\_Prepar}}(C,G) \quad (17)$$

$$\text{Steps}_{\text{Sim\_Flooding}}(C,G) = \text{Steps}_{\text{OIM}}(C,G) + \text{Steps}_{\text{Init\_map}}(C,G) + \\ + \text{Steps}_{\text{PCG}}(C,G) + \text{Steps}_{\text{Flooding}}(C,G) \quad (18)$$

$$\text{Steps}_{\text{Combined}}(C,G) = \text{Steps}_{\text{TreeConstr}}(C,G) + \text{Steps}_{\text{WordNET}}(C,G) + \\ + \sum_{i,j} \text{Steps}_{\text{ancest}}(C_i,G_j) + \text{Steps}_{\text{child}}(C_i,G_j) + \text{Steps}_{\text{leaves}}(C_i,G_j) \quad (19)$$

In the next section the calculation complexity of the different approaches is evaluated on multiple test cases.

4 Experiments

To estimate and analyze working cost i.e. number of run-time steps for different schemas more samples was implemented. As the reader could see in the previous chapter, calculation of the expected number of run-time steps is not an easy method especially for custom (e.g. real-life) scenarios. On the other hand computational costs of real schemas can also be estimated with that of some artificial schemas of the right size. Thus specific samples were created so that the required parameters e.g. average path length to the root, or to the leaves can be easily calculated. These schemas conform to strict rules as follows:

- every intern node (complex type having at least one attribute with a range complex type as well) has the same number of children (i.e. the same number of attributes with a complex range),
- every node has the same number of attributes (in other worlds, the summary of complex and simple attributes for every nodes is constant),
- all leaves of the tree can be found at the same level (in other words, the distance of complex types having only simple attributes from the root element is always the same) and
- every complex type has the same number of attached terms.

4.1 Implemented test schemas

Upon the rules described above the complexity of the used sample schemas can be described by the following parameters:

- the number of branches represents the number of children nodes for intern complex types,
- the number of deepness shows the distance of leaf nodes from the root,
- concept attribute is the number of attributes connected to complex types and
- terms depicts the number of associated terms of complex types.

The implemented test schemas have the following parameters, see table 1.

Table 1 Parameters of implemented test schemas

	Brances	Deepness	Attributes	Terms
Sample	1,33	1,375	5	4
Example1	2	2	7	5
Example2	3	4	9	6
Example3	5	5	12	8

A more practical representation of schemas is the graph representation. Figure 1 shows the graph of the test schema *Example1*. *Example2* and *Example3* are built up and can be presented similarly. The test data called *Sample* was implemented differently. It is not conformed to the strict rules described above but its small size facilitates to define the necessary parameters to the calculations. Furthermore this not "standardized" test data may also validate the usage of other artificial schema examples by showing expected values in run-time steps and actual run-time measurements. Figure 2 shows the graph representation of test schema called *Sample*.

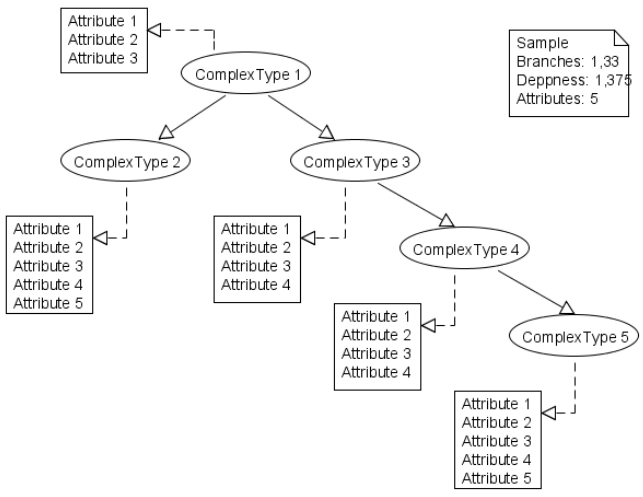

Fig. 2. Graph representation of test schema Sample

To be able to calculate the number of expected steps the following values were also approximated:

- number of different simple types and flooding iterations for approach similarity flooding and

- average number of children and leaves, and average length of path to the root and to the leaves for calculating the steps of the combined matcher (approach WordNet).

The test cases were defined as all possible combination of schema pairs. Including also the evaluation of the same schemas this means 10 test-cases from different complexity. The next sub-chapter presents the calculations detailed for the evaluation of schema Sample with test schema Sample. Please remember, that from the point of view of computational complexity and required run-time costs it is not important that the schema is actually compared with itself.

4.2 Detailed calculation of a given example

The calculation of similarity values of complex types for test schemas Sample (consisting of 5-5 complex types) means

$$\text{Steps}_{\text{NTA\_Prepar}}(C,G) = 5*5+5*5+5+5+5*4+5*4 = 100 \quad (20)$$

steps of preparation, and

$$\text{Steps}_{\text{NTA\_Perform}}(C,G) = 25+25*5*5+25*5*5 = 1275 \quad (21)$$

steps of similarity calculation for this estimated size of schemas.

Hence the number of overall steps of the NTA approach for these schemas is:

$$\text{Steps}_{\text{NTA\_Sample}}(C,G) = 100+1275 = 1375. \quad (22)$$

The next presented approach was the similarity flooding. The steps to create the OIM graph representation of both schemas is as follows:

$$\text{Steps}_{\text{OIM}}(C,G) = 2*\left(\begin{array}{l}\sum_i \|\text{Attr}(C_i)\| + \\ \sum_j \|\text{Attr}(G_j)\| + \|C\| + \|G\| + Y_C + Y_G + 3\end{array}\right) = \quad (23)$$

$$2*(5*5+5*5+5+5+6+6+3) = 150$$

where values and should have been estimated. These are data types e.g. string, boolean, integer, float, date and time, so the number of existing data types both in the global and the local schema and is estimated now with 6 for this sample.

The steps for the initial mapping is calculated as follows:

$$\text{Steps}_{\text{Init\_map}}(C,G) = \left(3+2\|C\|+2\sum_i \|\text{Attr}(C_i)\|+2Y_c\right)* \quad (24)$$

$$*\left(3+2\|G\|+2\sum_j \|\text{Attr}(G_j)\|+2Y_G\right) =$$

$$= (3+2*5+2*5*5+2*6)^2 = 5625$$

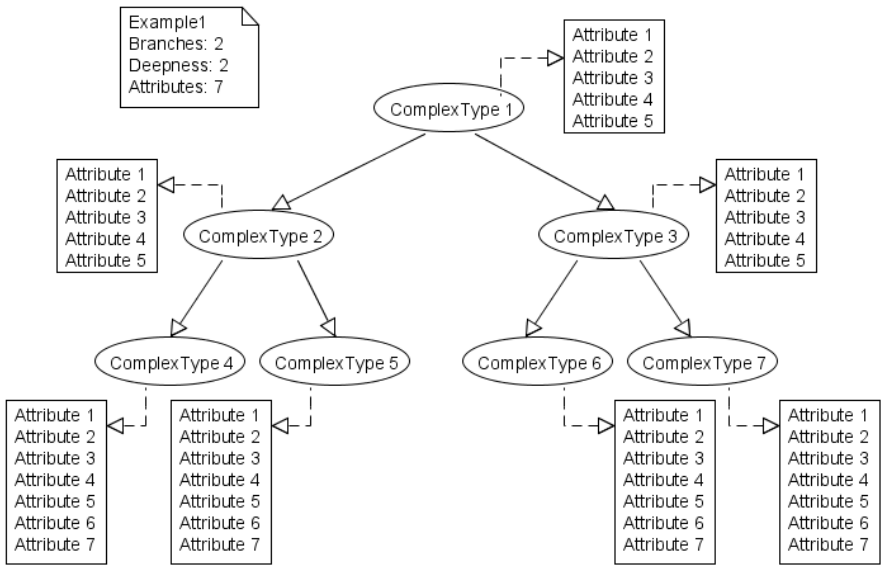

Fig. 1 Graph representation of Example1

The PCG graph creation costs

$$\begin{aligned}&\mathrm{Steps}_{\mathrm{PCG}}(C,G)=2\left(\|C\|+\sum_i\|\mathrm{Attr}(C_i)\|+Y_C\right)*\left(\|G\|+\sum_j\|\mathrm{Attr}(G_j)\|+Y_G\right)+\\&\sum_i\|\mathrm{Attr}(C_i)\|*\sum_i\|\mathrm{Attr}(G_j)\|+\sum_i\|\mathrm{Simple\_Attr}(C_i)\|*\\&*\sum_j\|\mathrm{Simple\_Attr}(G_j)\|=\\&=2(5+5*5+6)^2+(5*5)*(5*5)+21*21=3658\end{aligned}\tag{25}$$

where the overall number of simple type attributes was simply counted for the given test schema. (Note, that this counting was supported by appropriate implementation used mainly by the estimation of computational requirements of the WordNet approach, see later)

Based on the statements in the corresponding paper of the similarity flooding [12] 6-8 iterative steps are needed to evaluate final results at the flooding (this number is dependent on the selected convergence formula and input and output schema complexity) at schemas at these sizes. Hence we set the number of iterative steps to equal 7. Based on these estimated values and substituting the average number of the attributes of complex types and the number of complex types from our example, the computational cost of the flooding can be calculated as follows:

$$\begin{aligned}&\mathrm{Steps}_{\mathrm{Flooding}}(C,G)=\sum_{\mathrm{steps}}\mathrm{Nodes}_{\mathrm{PCG}}(C,G)=\\&=\sum_{\mathrm{steps}}\left(\begin{array}{l}2*\left(\|C\|+\sum_i\|\mathrm{Attr}(C_i)\|+Y_C\right)\\*\left(\|G\|+\sum_j\|\mathrm{Attr}(G_j)\|+Y_G\right)\end{array}\right)=7*\left(2*(5+5*5+6)^2\right)=18144\end{aligned}\tag{26}$$

The total number of steps can be calculated as follows:

$$\begin{aligned}&\mathrm{Steps}_{\mathrm{Sim\_Flooding}}(C,G)=\mathrm{Steps}_{\mathrm{OIM}}(C,G)+\mathrm{Steps}_{\mathrm{Init\_map}}(C,G)+\\&\mathrm{Steps}_{\mathrm{PCG}}(C,G)+\mathrm{Steps}_{\mathrm{Flooding}}(C,G)=\\&=150+5625+3658+18144=27577\end{aligned}\tag{27}$$

The last presented approach was the so called WordNet approach. Substituting the values of our example into the equations for the steps of the initial tree construction and retrieving WordNet we get the following results:

$$\begin{aligned}&\mathrm{Steps}_{\mathrm{TreeConstr}}(C,G)=\sum_i\|\mathrm{Attr}(C_i)\|+\sum_j\|\mathrm{Attr}(G_j)\|+\|C\|+\|G\|=\\&=5*5+5*5+5+5=60\end{aligned}\tag{28}$$

and

$$\begin{aligned}&\mathrm{Steps}_{\mathrm{WordNET}}(C,G)=\left(\|C\|+\sum_i\|\mathrm{Simple\_Attr}(C_i)\|\right)*\\&\left(\|G\|+\sum_j\|\mathrm{Simple\_Attr}(G_j)\|\right)==(5+21)^2=676\end{aligned}\tag{29}$$

The rest of the calculations are the evaluation of similarities in the ancestor, child and leaf context. Thus the average length of the path from the given node till the root should be estimated. Unfortunately this depends strongly on the given schemas and can not be estimated based on the number of complex types in the schema and average number of connected attributes per complex type. However some values can be assigned to our example. Suppose that the maximal depth in our schema forest is 4 and there is 1 complex type at distance 3, 2 complex types at distance 2 from the root element and 1 complex type is the root element itself. So the average length of path is estimated by 2,4 and the average for the expected number of steps for evaluating the ancestor similarity of two given complex types is:

$$\mathrm{Steps}_{\mathrm{ancest}}(C_i,G_j)\approx 2{,}4*2{,}4=5{,}76\tag{30}$$

To be able to estimate the computational complexity of the calculation of child context similarity the average number of children of complex types should be set. For both the local schema tree and the global schema tree this is exactly 5, because every complex type has 5 attributes in our example.
Hence the number of steps for calculating child context similarity is:

$$\begin{aligned}&\mathrm{Steps}_{\mathrm{child}}(C_i,G_j)\approx\left(\|\mathrm{child}(C_i)\|*\|\mathrm{child}(G_j)\|\right)+\\&+\left(\|\mathrm{child}(C_i)\|*\|\mathrm{child}(G_j)\|\right)*1{,}39\log_2\left(\|\mathrm{child}(C_i)\|*\|\mathrm{child}(G_j)\|\right)\\&=(5*5)+(5*5)*1{,}39\log_2(5*5)=186{,}374\end{aligned}\tag{31}$$

The leaf similarity can be calculated in a number of steps as follows:

$$\begin{aligned}&\mathrm{Steps}_{\mathrm{leaves}}(C_i,G_j)\approx\sum_{\|\mathrm{leaf}(C_i)\|\|\mathrm{leaf}(G_j)\|}\mathrm{length}(P_1)*\mathrm{length}(P_2)+\\&\|\mathrm{leaf}(C_i)\|\|\mathrm{leaf}(G_j)\|*1{,}39*\log_2\|\mathrm{leaf}(C_i)\|\|\mathrm{leaf}(G_j)\|=\\&=10{,}6^2*2{,}96^2+10{,}6^2 1{,}39*\log_2\left(10{,}6^2\right)=2049{,}858\end{aligned}\tag{32}$$

where $\|\mathrm{leaf}(C_i)\|$ and $\|\mathrm{leaf}(G_j)\|$ were estimated with the average number of leaves under the nodes in our sample tree, which is:

$$\|\mathrm{leaf}(C_i)\|=\|\mathrm{leaf}(G_j)\|\approx\frac{21+18+9+5}{5}=10{,}6\tag{33}$$

and the average length of the paths to the leaves is estimated with:

$$\begin{aligned}&\mathrm{length}(P_1)=\mathrm{length}(P_2)\approx\\&\approx\frac{2*(5+5+4+4+3)+3*(5+4+9)+4*(5+4)+5*5}{21+18+9+5}\approx 2{,}96\end{aligned}\tag{34}$$

.

These calculations were also facilitated by appropriate implementations also mentioned above of course.

Substituting the estimated number of steps of preparations and calculating ancestor, child and leaf context to each node into expression 1.47 the overall performance is approximated as follows:

$$\begin{aligned}&\mathrm{Steps}_{\mathrm{Combined}}(C,G)=\mathrm{Steps}_{\mathrm{TreeConstr}}(C,G)+\mathrm{Steps}_{\mathrm{WordNET}}(C,G)+\\&\sum_{i,j}\mathrm{Steps}_{\mathrm{ancest}}(C_i,G_j)+\mathrm{Steps}_{\mathrm{child}}(C_i,G_j)+\mathrm{Steps}_{\mathrm{leaves}}(C_i,G_j)=\\&60+676+5*5*(5{,}76+186{,}374+2049{,}858)=56786\end{aligned} \quad (35)$$

So comparing the results for the 3 different approaches we get the following order:

$$\begin{aligned}&(\mathrm{Steps}_{\mathrm{NTA\_Sample}}(C,G)=1375)<(\mathrm{Steps}_{\mathrm{SimFlooding\_Sample}}(C,G)=27577)<\\&<(\mathrm{Steps}_{\mathrm{Combined\_Sample}}(C,G)\approx 56786)\end{aligned} \quad (36)$$

where one can see, that the estimated cost of our approach called NTA is significantly smaller than the cost of other approaches.

After the estimation of the number of required computational steps the results were also validated by the actual measurement of steps by running the specific implementations. The measured numbers of steps are shown in Table 2.

Table 2 Measured number of run-time steps for *Sample*

NTA	Sim. Flood	WordNet
1375	18827	54256

As the reader can see the measured numbers are close to the estimated results validating our method.

Finally the connection between the number of required computational steps and actual run-time is evaluated. Unfortunately this test scenario – comparing test schema *Sample* with itself – was not complex enough to result in real differences among the run-time costs: the run times were close to zero (signaling only the sampling frequency of the time measuring function) in all cases. Therefore the differences in run-time will be evaluated in the case of more complicated test scenarios – see in Chapter 4.3 later.

4.3 Overall evaluation

The estimated number of run-time steps for every approach is presented in table 3.

Table 3 Estimated number of run-time steps

Test scenario	NTA	Sim. Flood	WordNet
Sample vs Sample	1375	19801	56981
Example1 vs Sample	2451	33663	95619
Example1 vs Example1	4396	57335	252207
Example2 vs Sample	59471	495365	5865024
Example1 vs Example2	91809	847019	12086636
Example2 vs Example2	1921843	16930913	610377358
Example3 vs Sample	2152256	20763658	361249806
Example1 vs Example3	3882655	35530560	609118736
Example2 vs Example3	81375634	1204698630	37622299563
Example3 vs Example3	3448208988	50469880392	2,31949E+12

To validate the correctness of our approximation method specific step counters were placed into the implementation of the algorithms. The number of steps measured during the execution is shown in table 4.

Table 4 Measured number of run-time steps

Test scenario	NTA	Sim. Flood	WordNet
Sample vs Sample	1375	18827	54256
Example1 vs Sample	2451	32837	113933
Example1 vs Example1	4369	57371	240700
Example2 vs Sample	59471	483215	5821611
Example1 vs Example2	91809	847037	12024888
Example2 vs Example2	1921843	16930931	622136464
Example3 vs Sample	2152256	20255808	380147160
Example1 vs Example3	3882655	65530578	790026787
Example2 vs Example3	81375634	N/A	N/A
Example3 vs Example3	3448208988	N/A	N/A

The measured values have successfully validated our approximation method. Not surprisingly by the approach NTA the same values were returned by the test, because neither average calculation nor any other approximations were applied there. But the measured values of other approaches are also close to our estimations. The “N/A” symbol stands for not successful execution of the given test scenarios. Because of the size of these scenarios and the higher complexity of algorithms they can not be executed on regular hardware configurations e.g. on personal computers or the run-time is enormous long.

Based on our calculations the order of the actual execution is also predictable. However without knowing the exact capacity of run-time configuration and execution of some simpler scenarios for achieving some reference measurements more accurate estimation of execution time is not possible. On the other hand, fully accurate run times are not need in most of the cases, only the correct order of magnitude is important. Our measured execution times are shown in table 5.

Table 5 Measured execution times of the tests (seconds)

Test scenario	NTA	Sim. Flood	WordNet
Sample vs Sample	0,03	0,06	0,08
Example1 vs Sample	0,03	0,06	0,08
Example1 vs Example1	0,03	0,08	0,11
Example2 vs Sample	0,08	0,63	1,00
Example1 vs Example2	0,08	1,11	1,95
Example2 vs Example2	0,92	25,94	98,23
Example3 vs Sample	3,56	41,91	91,88
Example1 vs Example3	2,52	206,48	159,06
Example2 vs Example3	40,52	N/A	N/A
Example3 vs Example3	2264,84	N/A	N/A

5 Short evaluation of accuracy

As already mentioned above by evaluating schema matching approaches both accuracy and required computational cost should be taken into account. Although the aim of this paper was a detailed analyzes of run-time cost the results of our accuracy measurements are also presented in table 6. As measurement methods and units the mostly used parameters were adapted from current literature [4, 12]. These are the followings:

- The precision returns a value comparing the number of correctly found pairs with the number of all proposed pairs. (Correctly found pairs are both part of the ideal matching and evaluated algorithm result.) The precision is calculated as follows:
  Precision = Correctly found pairs / All proposed pairs
- The recall returns a value comparing the number of correctly found pairs with the number of all existing pairs in the ideal matching. The recall is calculated as follows:
  Recall= Correctly found pairs/ All pairs of the ideal matching
- The F-measure is a combined indicator originated from the previous two:
  F-measure= 2*Precision*Recall/(Precision+Recall)

Table 6 Algorithm accuracy of different approaches

Scenarios:	Scenario 1			Scenario 2			Scenario 3		
Indicators	NTA	SF	WN	NTA	SF	WN	NTA	SF	WN
Precision	**0,8**	0,25	1	0,46	**0,53**	0,21	**0,55**	0,33	0,31
Recall	**1**	1	0,75	0,55	0,73	**0,82**	**0,86**	0,86	0,71
F-Measure	**0,89**	0,4	0,86	0,5	**0,61**	0,33	**0,67**	0,48	0,43

Our previously developed schema matching algorithm (called NTA) performed quite well comparing to others. Taking also into account its much lower computational requirement it clearly outperforms other approaches.

6 Conclusion

The computational cost of schema matching approaches were analyzed and estimated by different mathematical methods and techniques. This is useful by designing and comparing of schema matching solutions willing to be used in everyday work or at critical on-line systems. The presented approach is applicable for various kinds of matching approaches; typical computational tasks of linguistic, structural and combined matchers were all covered. As a demonstration, we have also successfully validated our method on three different approaches. Our experiments showed that the estimated values are correct, and the required run-time complexity and the order of magnitude of the actual execution times are also predictable based on our returned results.

References:

[1] I. Astrova, A. Kalja, A novel approach to mapping SQL relational schemata to OWL ontologies, *WSEAS Transactions on Information Science and Applications*, Vol. 3, Issue 11, 2006, pp. 2215-2222.

[2] Bergamaschi, S., S. Castano, and M. Vincini: Semantic Integration of Semistructured and Structured Data Sources, *SIGMOD Record*, Vol. 28, Issue 1, 1999, Pp. 54-59.

[3] V. C. Bhavsar, H. Boley, L. Yang, A weighted-tree similarity algorithm for multi-agent systems in E-business, *Computational Intelligence*, Vol. 20, Issue 4, pp. 584-602.

[4] Aida Boukottaya, Christine Vanoirbeek, Schema Matching for Transforming Structured Documents, *Proceedings of the 2005 ACM symposium on Document engineering*, 2005, pp. 101-110.

[5] D. Buttler, A Short Survey of Document Structure Similarity Algorithms, *Proceedings of the 5th international conference on internet computing*, 2004.

[6] Cognitive Science Laboratory, WordNet - a lexical database for the English language, at http://wordnet.princeton.edu/

[7] Hong-Hai Do, Erhard Rahm, Matching large schemas: Approaches and evaluation, *Information Systems*, Vol. 32, Issue 6, 2007, pp. 857-885.

[8] Buhwan Jeong, Daewon Lee, Hyunbo Cho, Jaewook Lee, A novel method for measuring semantic similarity for XML schema matching, *Expert Systems with Applications*, Vol. 34, Issue 3, 2008, pp. 1651-1658.

[9] J. Madhavan, P. A. Bernstein, E. Rahm, Generic Schema Matching with Cupid, *Proceedings of the 27th International Conference on Very Large Data Bases*, 2001, pp. 49-58.

[10] Martinek P., Szikora B, Detecting semantically related concepts in a SOA integration scenario, *Periodica Polytechnica*, 2008. – in press.

[11] Peter Martinek, Balazs Tothfalussy, Bela Szikora, Implementation of semantic services in enterprise application integration, *WSEAS Transactions on Computers*, Vol. 7, Issue 10, 2008, pp. 1658-1668.

[12] S. Melnik, H. Garcia-Molina, E. Rahm, Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching, *Proceedings of the 18th International Conference on Data Engineering*, 2002, pp. 117-128.

[13] Richi Nayak, Wina Iryadi, XML schema clustering with semantic and hierarchical similarity measures, *Knowledge-Based Systems*, Vol. 20, Issue 4, 2007, pp. 336-349.

[14] Palopoli, L. G. Terracina, and D. Ursino: The System DIKE: Towards the Semi-Automatic Synthesis of Cooperative Information Systems and

Data Warehouses, *Lecture Notes in Computer Science*, Vol. 2282, 2002, pp. 228-276.

[15] Khalid Saleem, Zohra Bellahsene, Ela Hunt, Performance Oriented Schema Matching, *Lecture Notes in Computer Science*, Vol. 4653, 2007, pp.844-853.

[16] A. Salguero, et. al, Ontology based framework for data integration, *WSEAS Transactions on Information Science and Applications*, Vol. 5, Issue 6, 2008, Pp. 953-962.

[17] H. Quyet Thang, V. Sy Nam, XML Schema Automatic Matching Solution, *International Journal of Computer Systems Science and Engineering*, Vol 4., Num. 1, pp. 68-74.

[18] Yu J., Zhou G., SG: A structure based Web Services matching framework, *WSEAS Transactions on Information Science and Applications*, Vol. 4, Issue 4, 2007, Pp. 669-673**.**

[19] Wu X., Feng J., A framework and implementation of information content reasoning in a database, *WSEAS Transactions on Information Science and Applications*, Vol. 6, Issue 4, 2009, pp. 579-588.