International Journal of Computational Methods
© World Scientific Publishing Company

TWO ALGORITHMS FOR FAST INCREMENTAL TRANSITIVE CLOSURE OF SPARSE FUZZY BINARY RELATIONS*

MANOLIS WALLACE
Department of Computer Science,
University of Indianapolis, Athens Campus,
9, Ipitou Str., Syntagma, 105 57, Athens, Greece
wallace@uindy.gr
http://cs.uindy.gr

STEFANOS KOLLIAS
School of Electrical and Computer Engineering,
National Technical University of Athens,
9, Iroon Polytechniou Str., 157 73 Zographou, Athens, Greece
stefanos@cs.ntua.gr
http://www.image.ntua.gr

Received (9 December 2004)
Revised (26 June 2006)
Accepted (13 October 2006)

Sparse fuzzy ordering and partial ordering relations have recently become of great importance in the field of knowledge systems. As the size of the relations utilized in such a framework is extremely large, efficient algorithms are needed for their handling. More specifically, when a part of such a relation is updated, the property of transitivity needs to be re-established in timely manner, as the knowledge system often becomes unusable until this process is completed. In this paper we propose two algorithms for the incremental update of fuzzy transitive relations. The first one focuses on the incremental update of a part of an already transitive relation, while the other tackles the complete transitive closure of any relation. For the average sparse relation, both of the proposed algorithms have considerably smaller computational complexity than the classical approach, which we also prove experimentally via application to real life relations of this type.

Keywords: transitive closure; complexity; sparse matrix; partial ordering relations

1. Introduction

Quite recently, through the developments of ontological representation and storage of knowledge [Maedche (2002)], the generation and handling of large but sparse binary relations has become an issue that finds application in real life systems. In most

*This work has been partially funded by the HERACLETUS project of the Greek Ministry of Education.

cases, the semantic meaning of these relations implies the property of transitivity. If we consider, for example, the specialization relation, entity mammal will be linked to entity human and entity human to entity man. The link between mammal and man is implied. The size of the relations is such that these implications cannot be calculated on-line; the relation needs to be available in an already transitive form if it is to be used by a knowledge system that aims to respond to user requests in a timely manner.

Moreover, in parallel to the developments in the field of ontologies, the utilization of fuzzy binary relations for the representation of knowledge, towards more intelligent and semantics based information and multimedia retrieval [Avrithis et. al (2001)], has also started to be investigated. More information on the properties of these relations and the algorithms that utilize them can be acquired from [Akrivas et. al (2002)][Wallace et. al (2003a)][Wallace et. al (2003b)][Wallace et. al (2002)][Avrithis et. al (2003)]. Similar arguments to those of the previous case lead to the conclusion the the transitivity of the relations is required.

The transitive property of binary relations, due to its physical meaning, is closely related to the study of graphs. In that framework, transitive closure of a relation is equivalent to the detection of the pairs of vertices that are either directly connected or connected via some path. Thus, the majority of existing literature on transitive closure and has focused mainly on the cases of unweighed [Warshall (1962)][Warren (1975)][Purdom (1970)][Thorelli (1966)][Baker (1962)] and usually undirected graphs [Seidel (1992)].

Works that focus on the I/O complexity of the algorithms also exist, but they too are limited to the Boolean case [Ullman (1991)]. Some of these algorithms may extended rather easily to support for weights on the edges, thus becoming able to handle the case of fuzzy relations as well. Still, we have not been able to locate any transitive closure algorithm that considers the case of sparse relations; this will be the main focus of the algorithms presented in this paper.

Given the size of the considered binary relation, their representation is as great a problem as that of their handling. Specifically, the memory required for the storage of an $n \times n$ array is prohibitive, as n is the count semantic entities that are known to the system and is often close to $n \simeq 70000$; we may overcome this problem with the utilization of a more compact representation format, since utilized relations are typically very sparse; this has had a small expense of computational time required to access a specific element. Still, the two following issues, related to the theory and practice of transitive closure of fuzzy ordering binary relations will still have to be tackled:

- Most retrieval and document analysis algorithms that utilize relational knowledge assume that the fuzzy binary relation they accept as input is transitive in some form.Thus, the transitive closure of the $n \times n$ relation has to be computed before the system is made operable for the first time.
- Relations that represent human knowledge are typically constructed using

a trial and error strategy, which means that small adjustments are often made, thus locally disturbing the transitivity of the relation. A new time consuming transitive closure after each adjustment would make the application of the trial and error strategy impossible, thus calling for a method for incremental adjustment of the property of transitivity.

Both of these issues are tackled using the sparse representation model and the algorithms presented in this paper. Specifically, in section 2 we start by discussing the compact representation model chosen for the relations, as the properties of this data model directly affect the properties of the algorithms to follow. In the same section we also discuss the properties of the classical approach to transitive closure and explain what we mean when we refer to sparse relations. Continuing, in section 3 we discuss the properties of the classical transitive closure algorithm, when combined with the representation model presented in section 2. In sections 4 and 5 we present the two algorithms that we propose. Specifically, in section 4 present our approach for incremental update of a binary relation and discuss its storage and computational merits and in section 5 we extend our discussion to explain how this can be utilized to simplify the complete transitive closure of a relation as well. Section 6 lists a few indicative experimental results and section 7 lists our concluding remarks.

2. Preliminaries

2.1. *Data model*

A fuzzy binary relation defined on a set S containing n distinct elements may be represented as a square matrix of dimension n. The physical storage of such an array requires the representation of n^2 different decimal values, which for large numbers of n is prohibitive for practical implementation. On the other hand, in such a representation access to a specific element of the array has a computational complexity of $O(1)$, as the position of the element directly specifies its position in storage as well, with the utilization of a formula of the form

$$M = (i-1) * n + j \tag{1}$$

where (i, j) is the position of the element in the matrix and M is its actual position in storage. As explained in section 1, the utilization of such a representation is not always possible. In the cases that the relation is known to be sparse, i.e. only a small subset of the elements of the corresponding array are non zero, then a sparse array implementation can be used to overcome the storage problem.

Classical sparse array implementation utilizes linked lists to represent elements, thus raising the computational complexity of accessing a specific element to $O(n)$. In this case, although the storage requirements are much smaller, the representation model remains inapplicable for time critical applications where operations utilizing a binary relation have to be performed before the system response is determined, and the number n is large.

The representation model proposed in order to overcome these limitations is as follows: a binary relation is represented using two sorted vectors. The first vector is sorted according to index i, and in case of identical row positions, column position j is utilized, and vice versa for the second vector.

$$\begin{bmatrix} & (1,2) & & & (1,5) & (1,6) \\ (2,1) & & & (2,4) & (2,5) & \\ & (3,2) & & & & \\ & & & (4,4) & & (4,6) \\ (5,1) & & & (5,4) & & \end{bmatrix}$$

The above array, for example, is represented using the following vectors:
[(1,2), (1,5), (1,6), (2,1), (2,4), (2,5), (3,2), (4,4), (4,6), (5,1), (5,4)]^T
[(2,1), (5,1), (1,2), (3,2), (2,4), (4,4), (5,4), (1,5), (2,5), (1,6), (4,6)]^T

This representation model preserves the storage merits of the classical sparse matrix implementation with the linked lists. Moreover, access time for a specific element, row or column has a computational complexity of $O(logn)$, when utilizing a divide and conquer approach, which is much better than the linear complexity of the classical approach.

2.2. *Classical transitive closure*

A complete transitive closure of a fuzzy binary relation may be performed utilizing the following methodology [Klir and Yan (1995)]:

In the general case, the transitive closure $Tr^t(r)$ of relation r, given some t-norm, can be calculated as

$$Tr^t(r) = \bigcup_{f=1}^{\infty} r^f \tag{2}$$

$$r^{f+1} = r^f \circ^t r \tag{3}$$

$$r^1 = r \tag{4}$$

In equation 2, as well as in all subsequent equations in this paper, the classical fuzzy co-norm max is assumed. Given that the universe S is a finite set, i.e. that $|S| = n$, equation 2 can be rewritten as :

$$Tr^t(r) = \bigcup_{f=1}^{n-1} r^f \tag{5}$$

Moreover, in order to avoid some steps of the described recursion and lower the computational complexity, equaiton 3 can be rewritten as [Klir and Yan (1995)]:

$$r^{2f} = r^f \circ^t r^f \tag{6}$$

Finally, in order to avoid performing the costly operation of relation composition more times than needed, the process may stop before reaching $f \geq n-1$, if it is found that $r^{2f} = r^f$, as it is then easily proven that $Tr^t(r) = r^f$.

As a special case, when relation r is reflective, it is proven that the transitive closure is given by equation

$$Tr^t(r) = r^{n-1} \tag{7}$$

Lemma: In the case of complete representation, the calculation of the composition of two relations has a computational complexity of $O(n^3)$.

Proof: An element r^2_{ij} of the composition is calculated as

$$r^2_{ij} = \bigcup_{c=1}^{n} r_{ic} \cap_t r_{cj} \tag{8}$$

This is an $O(n)$ operation. As the output of the composition has a dimension of $n \times n$, n^2 such operations need to be performed, thus concluding in an overall complexity of $O(n^3)$.

$\square$

Easily we have the following

Lemma: The transitive closure has a computational complexity of $O(n^3 logn)$, for both reflective and non reflective binary relations.

Proof: In the reflective case, equation 7 can be used to calculate the transitive closure of the relation. Utilizing equation 6 we can calculate r^{n-1} in $O(logn)$ compositions. Each one of them, as proven above, has a complexity of $O(n^3)$, thus summing up to $O(n^3 logn)$

In the non reflective case, combining equation 5 and equation 6 we can see that we need $O(logn)$ additions and compositions. An addition has a complexity of $O(n^2)$ and a composition, as shown above $O(n^3)$. Thus, the overall complexity is again

$$O(logn) \cdot (O(n^3) + O(n^2)) = O(n^3 logn) \tag{9}$$

$\square$

An overall complexity of $O(n^3)$ can also be achieved through small modifications in the sequence in which elements of the relation are examined and updated. Unfortunately, it is not possible to alter this approach so that complexity can be further reduced in the case of sparse relations which is the topic of the discussion for this work.

2.3. *Sparse relations*

Binary relations can be used to model numerous aspects of the real world. Depending on the case, the considered relations may display various formal mathematical properties, such as associativity, reflexivity, transitivity of one form or another and so on. The specification of each one of these properties is an objective task, which makes it feasible to discriminate between the different types of relations and handle them in different ways.

On the other hand, some subjective properties are often important in real life relations. The most important among them is sparseness; sparse relations, although having the same objective properties as their dense counterparts, are "best" handled using totally different approaches. By "best" we do not refer to the validity of the methodologies applied, as their results are identical, but rather to their efficiency as far as storage and computational resources needed are concerned.

As sparse relations play an important role in various fields, specialized data models and corresponding algorithms have been developed just for them; one of the most important problems with such data models and algorithms is their evaluation. Traditionally, the efficiency of a data structure or algorithm is measured using its storage and computational complexity, which is closely related to its operation in the worst possible case. This is of course inapplicable for the case of models and algorithms that have been designed for the sparse case, as, by definition, the notion of worst case is contradictory to that of sparseness.

Thus, in order to make the evaluation possible, we typically refer to the performance in the average, rather than the worst case scenario. This, in turn, demands that we can formally define the statistical properties of the average case. In this work, driven by the statistical characteristics of sparse relations appearing in the field of ontologies, we define the average case for sparse relations as follows:

Definition: We will say that a relation has logarithmic density when it contains $O(n)$ non zero rows and $O(n)$ non zero columns with the count of non zero elements contained in a non zero row or column being proportional to the logarithm of the count of all the elements in the relation.

In such a relation we have $O(\log n)$ non zero elements in each non zero row and column and $O(n \log n)$ non zero elements in the relation overall. Since this type of sparseness is typical for the type of relations considered herein, we will refer to the case of relations with logarithmic density as the typical or average case.

3. Classical transitive closure of sparse binary relations

As we have already mentioned, the characteristics of the representation model utilized reflect on the characteristics of the applied algorithms as well. Thus, when it comes to the classical algorithm of transitive closure, the following hold:

Theorem: The complete transitive closure is achieved with computational complexity $O(n^2 logn)$ for relations with logarithmic density and $O(n^3 log^2 n)$ for dense relations when following the classical transitive closure methodology.

Proof: During a step of relation composition in the compact representation case, if row i and column j exist in the relation, i.e. if they have at least one non zero element, they are retrieved. As already explained, retrieving a row or column has a complexity of $O(logn)$. Continuing, the corresponding element r^2_{ij} is calculated as

$$r^2_{ij} = \bigcup_{r_{ic} \in row \vee r_{cj} \in col} r_{ic} \cap_t r_{cj} \tag{10}$$

As the row and column are both available in a sorted by index form, this is an $O(k_i^{row} + k_j^{col})$ operation, where k_i^{row} is the count of elements in row i and k_j^{col} is the count of elements in column j. For a single composition $a \cdot b$ such operations will be performed, where a is the count of non zero rows and b is the count of non zero columns of the relation. Overall, the complexity for a single composition is

$$2 \cdot O(logn) + O(a \cdot b) \cdot [O(k_i^{row} + k_j^{col}) + O(logn)]$$

where the last $O(logn)$ factor corresponds to the complexity of the insertion of an element to the output relation.

In the average case for sparse relations, a small percentage of the rows and columns will be non zero. Of course, although this may affect greatly the execution time required, it does not alter the complexity, as

$$O(a \cdot b) = O((p_a \cdot n) \cdot (p_b \cdot n)) = O((p_a \cdot p_b) \cdot n^2)) = O(n^2) \tag{11}$$

where p_a and p_b is the percentage of non zero rows and columns respectively. As far as the count of elements contained in a non zero row or column is concerned, this is assumed to be proportional to the logarithm of the count of all the elements in the relation. Thus $O(k_i^{row} + k_j^{col})$ is replaced by $O(logn)$. Overall, this leads to a complexity of

$$2 \cdot O(logn) + O(n^2) \cdot [O(logn) + O(logn)] = O(n^2 logn) \tag{12}$$

for the composition.

In the worst case, all the elements of the relation exist. In that case, $a = b = k_i^{row} = k_j^{col} = n$, and thus the complexity of the composition becomes

$$2 \cdot O(logn) + O(n^2) \cdot [O(n) + O(logn)] = O(n^3) \tag{13}$$

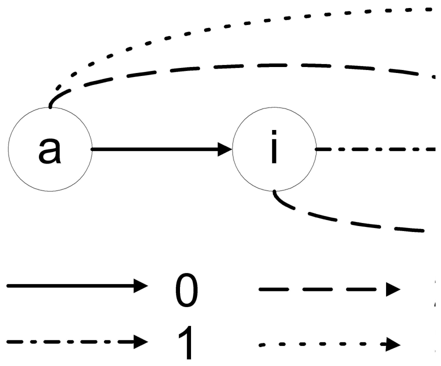
Considering the $O(logn)$ compositions are required for the transitive closure according to equation 6, it is easy to calculate that in the average case the complete transitive closure has a computational complexity of $O(n^2 log^2 n)$ and in the worst case $O(n^3 logn)$.

□

Proving that in the average sparse case the classical algorithm performs better if the proposed compact representation is used, compared to the complete representation, is rather trivial. It can be accomplished, by example, by showing that

$$lim_{x \to +\infty} \frac{x^3 logx}{x^2 log^2 x} = +\infty \tag{14}$$

Fig. 1. Graphical representation of the incremental update of the transitive relation.



As far as the storage requirements are concerned, the existence of two copies of the relation during the execution of the algorithm is required, for both compact and complete representations; the effect of this is of course much more intense in the case of complete representation, as much more space is required.

4. Incremental closure of fuzzy binary relations

A major disadvantage of the utilization of the classical transitive closure methodologies is that when for some reason an element of the relation is updated, or when a new element is inserted into the global set S, thus locally disturbing transitivity, a computationally demanding operation needs to be performed in order to assure the conservation of the transitivity property.

Theorem: When a new element is inserted into the relation or an element of the relation is updated, an $O(n^3 logn)$ operation has to be performed in order to assure that the relation remains transitive in the worst case, if following a self-composition approach. In the case of relations with logarithmic density the complexity is reduced to $O(n^2 logn)$.

Proof: Let r be a transitive relation. In Fig. 1 elements of r are represented using continuous lines of type 0. Let as suppose that r_{ij} is inserted in the relation, or that its value is augmented. In Fig. 1 the update is represented using dash - dot lines of type 1. Then, we can no longer assume that relation r is transitive.

After one self-composition, the ancestor a of i is linked to j and i is linked to the descendant d of j. In Fig. 1 this is represented using dashed lines of type 2. Finally, after one more self-composition, a is linked to b. In Fig. 1 this is represented using doted lines of type 3.

As r was initially assumed transitive, two compositions will always be enough to assure transitivity. Thus, the complexity of the operation that re-establishes transitivity is the same as that of the composition: $O(n^3)$ in the worst case and $O(n^2 logn)$ in the average case.

□

Having observed in Fig. 1 the way transitivity is achieved after a single element has been altered, we can design an algorithm for incremental update of transitive relations that has a considerably smaller computational complexity. Specifically, we may design an algorithm that will only focus on the changes that the complete transitive closure brings upon the relation after the update of the single element.

For any relation r this can be achieved with the following steps:

Algorithm:

(1) Identify the fuzzy set A of ancestors of i. Degrees in A are determined as

$$A(a) = r_{ai}, a \in S \tag{15}$$

(2) Identify the fuzzy set D of descendants of j. Degrees in D are determined as

$$D(d) = r_{jd}, d \in S \tag{16}$$

(3) For each element a appearing in A assign

$$r_{aj} = r_{aj} \cup (r_{ai} \cap_t r_{ij}) \tag{17}$$

(4) For each element d appearing in D assign

$$r_{id} = r_{id} \cup (r_{ij} \cap_t r_{jd}) \tag{18}$$

(5) For each element a appearing in A and d appearing in D assign

$$r_{ad} = r_{ad} \cup (r_{ai} \cap_t r_{ij} \cap_t r_{jd}) \tag{19}$$

If relation r is reflective, then $r(i,i) = 1$ and $r(j,j) = 1$ and thus $A(i) = 1$ and $D(j) = 1$. In this case, the above process can be simplified by omitting steps 3 and 4.

Theorem: The computational complexity of the incremental update algorithm is $O(n^2 logn)$ in the worst case and $O(log^3 n)$ in the case of relations with logarithmic density, for both reflective and non reflective relations.

Proof: We have already established that the complexity of steps 1 and 2 is $O(logn)$. The complexity of steps 3 and 4 is $O(k_i^{row}) \cdot O(logn)$ and $O(k_j^{col}) \cdot O(logn)$, respectively, and the complexity of step 5 is $O(k_i^{row} \cdot k_j^{col}) \cdot O(logn)$.

In the average case $k_j^{col} = O(logn)$, and thus the overall complexity is

$$2 \cdot O(logn) + 2 \cdot O(log^2 n) + O(log^3 n) = O(log^3 n) \tag{20}$$

In the worst case $k_j^{col} = n$, and thus the overall complexity is

$$2 \cdot O(logn) + 2 \cdot O(nlogn) + O(n^2 logn) = O(n^2 logn) \tag{21}$$

Ignoring the influence of steps 3 and 4 in the above calculations does not alter the overall complexity, as in all cases some other step contributes more to it. Thus, the same complexity holds for the reflective case as well.

□

4.1. *Comparisons*

Comparing the traditional and the incremental approaches to assuring transitivity, in the case of a relation that is already transitive and is undergoing a small change is rather easy. In all cases the incremental approach presented herein is superior:

- In the average case it has a complexity of $O(log^3 n)$ compared to a complexity of $O(n^2 logn)$ for the traditional approach.
- In the worst case it has a complexity of $O(n^2 logn)$ compared to a complexity of $O(n^3)$ for the traditional approach.

Moreover, the proposed approach is more efficient in means of storage requirements as well, as no copy of the relation needs to be created in memory.

5. Complete transitive closure of fuzzy binary relations

The aforementioned algorithm for incremental update of transitive fuzzy binary relations easily leads to the design of an algorithm for a complete transitive closure of a relation as well.

Algorithm:

```
(1) Create an empty binary relation $r'$.
(2) For each element $r_{ij}$ in the initial relation $r$
    (a) Assign $r'_{ij} = r'_{ij} \cup r_{ij}$
    (b) Run the incremental update algorithm on relation $r'$ with parameters $i$ and $j$.
```

When the algorithm terminates we have $Tr^t(r) = r'$.

Theorem: The computational complexity of proposed complete transitive closure algorithm is $O(nlog^4 n)$ in the average case and $O(n^4 logn)$ in the worst case.

Proof: Step 1 is obviously completed in an $O(1)$ operation. Step 2a is executed in an $O(logn)$ operation, while the complexity of step 2b is as described in the previous section. Step 2 is executed as many times, as is the count of elements in relation r.

In the worst case, the count of elements in r is n^2, and step 2b has a complexity of $O(n^2 logn)$, thus resulting in

$$2 \cdot O(1) + O(n^2) \cdot (O(logn) + O(n^2 logn)) = O(n^4 logn) \tag{22}$$

In the average case, there are $O(n)$ rows in r, each one containing $O(logn)$ elements, thus resulting in $O(nlogn)$ elements in r, and the complexity of step 2b is $O(log^2n)$. Thus, the overall complexity is

$$2 \cdot O(1) + O(nlogn) \cdot (O(logn) + O(log^3n)) = O(nlog^4n) \tag{23}$$

□

5.1. *Comparisons*

When it comes to a complete transitive closure, the comparison of the proposed approach to the classical one is a little more complicated than the corresponding comparison for the incremental update.

Clearly, in the dense / worst case, the classical approach is better, having a complexity of $O(n^3logn)$, compared to a complexity of $O(n^4logn)$ for the proposed approach.

In the average case, on the other hand, the proposed approach is greatly superior, having a complexity of $O(nlog^4n)$; proving that

$$O(n^3logn) > O(n^2) > O(nlog^4n)$$

is again trivial and can be achieved, for example, by proving that

$$lim_{x\to+\infty}\frac{x^2}{xlog^4x} = +\infty \tag{24}$$

Thus, comparison of the two approaches depends on the validity of the assumptions made concerning the average case. The main assumption made is that considerably less non zero elements exist in the average case; $O(nlogn)$ instead of $O(n^2)$. In the case of ontological relational knowledge, this assumption holds.

In addition to the enhanced computational complexity, the proposed approach to complete transitive closure of fuzzy binary relations also has the following merits:

- In the classical approach, a composition of the relation, and thus an operation of complexity $O(n^2logn)$ in the average case, has to be performed before the decision to stop the algorithm is taken. This is true even when the relation is initially transitive, thus requiring no adjustment. In the proposed approach, in the same situation the algorithm would terminate in $O(nlog^4n)$.
- In the classical approach, the relation is not transitive until the operation terminates. Thus, in the case of a very large number n, a very long time has to be spent before the relation becomes usable by algorithms that assume transitivity. On the contrary, relation r' is transitive after each step when utilizing the proposed complete transitive approach, and thus algorithms that assume it to be transitive can be applied to it before the whole operation is completed.
- Due to the recursive form of the classical approach, the termination point greatly depends on the "depth" of the relation, i.e. on the count of vertices

of the longest path. On the contrary, the proposed approach assures transitivity in one – pass and the count of required steps is known beforehand. Thus, an indication of the progress of the process is available on-line, if we want it, in the form of percent completed.

- The classical approach requires that two copies of the relation exist in the memory at the same time, thus doubling the storage requirements of the algorithm. On the other hand, space for a single relation is enough for the execution of the proposed approach; two relations exist, but as elements are added in one, they are removed from the other.
- The proposed approach is not affected by cycles in the relation, i.e. it does not require the relation to be ordering. This is is not a property shared by all transitive closure algorithms [Ullman (1991)].

6. Experimental results

The proposed methodology for transitive closure, as well as the classical approach, have both been implemented using the Java programming language. Moreover, a fuzzy binary relation representing all concepts and semantic entities in the English language has been prepared, utilizing WordNet as a source. This relation has a dimension of $n \simeq 70000$ and contains approximately 90000 non zero elements. The elements of the relation have been generated using the WordNet lexical relations. The complete (not compact) representation of this relation was not possible, as it would require storing approximately 5 billion double precision numbers; with 8 bytes allocated for each double precision number in Java, this means that 40Gb of RAM would be required just for the storage of one copy of the relation.

Utilizing the proposed compact representation form, a transitive closure of the relation has been attempted, using both the proposed and the classical approach; the t-norm used for the transitive closure was the min. The execution time for the classical approach was a little over five days. In this time, five compositions were completed. At this step the process was manually interrupted, without yet having reached assured closure (The two last copies of the relation were not identical). The execution time for the proposed approach was 6.45 seconds; in this time the transitive closure was completed.

Execution of both algorithms was performed in identical environments; the same computer was utilized in both cases (PIII 900 running W2K as operating system), and the computer was in both cases dedicated to the execution of the transitive closure algorithm.

7. Conclusions

In this paper, after proposing a compact representation model for sparse binary relations that allows for logarithmic access a row, a column or an element, we presented an algorithm for incremental update of transitive binary relations; we proved that this method outperforms the classical approach of performing a complete transitive closure in both the worst and the average case. Continuing, we have described an algorithm for transitive closure of any binary relation that relies on the above incremental methodology. We have proven that this method greatly outperforms the classical approach for the average sparse relation, achieving a computational complexity that is below $O(n^2)$. Finally, we have implemented the traditional and proposed methodologies and we have tested them on a real life fuzzy binary relation. The difference in execution time was immense, proving the efficiency of the proposed algorithm.

The results of this work can be directly applied in the fields of knowledge based systems and intelligent information and multimedia retrieval, as they allow for the efficient handling of large sparse relations. Other fields and applications that require the transitive closure of binary relations, fuzzy or not, may benefit from this work if the relations they deal with conform to the assumption of sparseness.

Acknowledgement

The authors wish to acknowledge the contribution of Dr. Yannis Avrithis, Senior Researcher of the Image, Video and Multimedia Systems Laboratory, in the design and development of the compact relation representation modules.

References

Akrivas, G., Wallace, M., Andreou, G., Stamou, G. and Kollias, S., "Context - Sensitive Semantic Query Expansion", ICAIS, Divnomorskoe, Russia, 2002.

Avrithis Y. and Stamou G.: "FAETHON: Unified Intelligent Access to Heterogeneous Audiovisual Content", VLBV, Athens, Greece, 2001.

Avrithis, Y., Stamou, G, Wallace, M., Marques, F., Salembier, P., Giro, X., Haas, W., Vallant, H. and Zufferey, M., "Unified Access to Heterogeneous Audiovisual Archives", I-KNOW, Graz, Austria, 2003.

Purdom, P., "A transitive closure algorithm", BIT vol. 10 pp. 76-94, 1970.

Baker, J.J., "A note on multiplying Boolean matrices", Comm. ACM vol. 5 no. 2, pp. 102, 1962.

Klir, G., Yuan, B., "Fuzzy Sets and Fuzzy Logic: Theory and Applications", Prentice Hall, 1995.

Maedche, A.,Motik, B., Silva, N., Volz, R.," MAFRA - An Ontology MApping FRAmework in the Context of the SemanticWeb". Workshop on Ontology Transformation, ECAI, 2002.

Seidel, R., "On the All-Pairs-Shortest-Path problem in unweighted undirected graphs", J. Computer & System Sciences vol. 51, pp. 400-403, 1995.

Thorelli, L.-E., "An algorithm for computing all paths in a graph", BIT vol. 6 pp. 347-349, 1966.

Ullman, J.D., Yannakakis, M., "The Input/Output Complexity of Transitive Closure", Annals of Mathematics and Artificial Intelligence, vol. pp. 331-360, 1991.

Wallace, M., Akrivas, G., Mylonas, P., Avrithis, Y., Kollias, S. "Using context and fuzzy relations to interpret multimedia content", CBMI, IRISA, Rennes, France, 2003.

Wallace, M., Akrivas, G. and Stamou, G., "Automatic Thematic Categorization of Documents Using a Fuzzy Taxonomy and Fuzzy Hierarchical Clustering", FUZZ-IEEE, St. Louis, MO, USA, 2003.

Wallace, M., Akrivas, G., Stamou, G. and Kollias, S., "Representation of user preferences and adaptation to context in multimedia content – based retrieval", Workshop on Multimedia Semantics, SOFSEM, Milovy, Czech Republic, 2002.

Wallace, M., Kollias, S., "Computationally efficient incremental transitive closure of sparse fuzzy binary relations", submitted to the IEEE International Conference of Fuzzy Systems (FUZZ-IEEE) in January 2004.

Warren, H.S., "A modification of Warshall's algorithm for the transitive closure of binary relations", Comm. ACM vol. 18 no. 4 pp. 218-220, 1975.

Warshall, S., "A theorem on Boolean matrices", J. ACM vol. 9 no. 1, pp. 11-12, 1962.