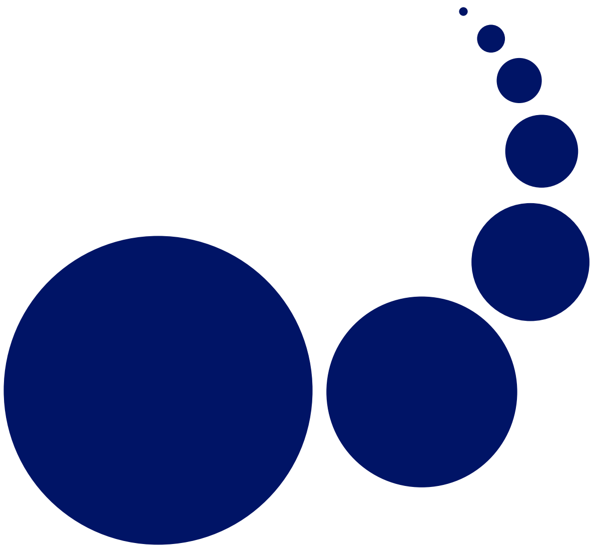
SCALABLE COMPUTING
Practice and Experience

Special Issue: Distributed Intelligent Systems

Editors: Maria Ganzha, Marcin Paprzycki



Volume 9, Number 1, March 2008

ISSN 1895-1767

SWPS

SUBSCRIPTION INFORMATION: please visit http://www.scpe.org

Scalable Computing: Practice and Experience

Volume 9, Number 1, March 2008

TABLE OF CONTENTS

Scalable Computing: Practice and Experience
Volume 9, Number 1, p. i. http://www.scpe.org
ISSN 1895-1767
© 2008 SCPE

INTRODUCTION TO THE SPECIAL ISSUE: DISTRIBUTED INTELLIGENT SYSTEMS

The first edition of the International Symposium on Intelligent and Distributed Computing—IDC'2007, took place on October 18–20, 2007 in Craiova, Romania. According to these who participated, Fall was beautiful in Craiova—as can be seen on the photograph below.

While the Symposium itself covered a large spectrum of issues involved in *Intelligent Computing,* for this Special Issue we have selected these that match the specific focus of the SCPE. Therefore, out of 33 papers published in Symposium Proceedings we have selected the following six.

In *Studying SVM Method's Scalability Using Text Documents,* D. Morariu, M. Vinţan, and L. Vinţan consider application of the SVN method to the case of automatic classification of large sets of text documents. They pay particular attention to the scalability of the approach and propose a method that allows working with very large data sets without increasing exponentially the training time and without significantly decreasing the classification accuracy.

Next, in *Learning Objects' Architecture and Indexing in WELSA Adaptive Educational System,* E. Popescu, C. Bădică, and P. Trigano propose ways of organizing learning material in an adaptive educational hypermedia system. They are particularly interested in use of instructional metadata to facilitate detection of student learning style and application of adaptation techniques. Implementation of the proposed approach in the WELSA system is presented and discussed.

Paper *A Link-cluster Route Discovery Protocol for Ad Hoc Networks,* by D. Bein, A. K. Datta and S. Yellenki introduces a route discovery algorithm for MANETs. This algorithm attempts at minimizing number of clusterheads and gateway nodes to avoid storing redundant data. Furthermore, it adapts to arbitrary movement of nodes, and joining and/or leaving of existent nodes.

Focus of the next contribution entitled *The Dilemma of Trust: a Social Network Based Approach,* by V. Carchiolo, A. Longheu, M. Malgeri, G. Mangioni, and V. Nicosia studies how social networks can be usefully exploited in addressing the question of trust. In their approach, authors try to reproduce real-life behavior of humans when they establish and maintain trust relationships and utilize this approach to build an effective and efficient model for trust management.

Next, in *APP: Agent Planning Package,* S. Tošić, M. Radovanović and M. Ivanović introduce APP, a domain independent Agent Planning Package, which facilitates creation of intelligent agents. Within the APP, agents can generate plans for their operating environment, as well as execute them and supervise the execution.

Finally, in *Knowledge Processing for Web Search—an Integrated Model and Experiments,* P. Gurský, T. Horváth, J. Jirásek, R. Novotný, J. Pribolová, V. Vaneková and P. Vojtáš propose a model of a middleware system enabling personalized web search for users with different preferences. Their approach integrates inductive and deductive tasks as well as fuzzy sets and logic to find user preferences and consequently best objects. The proposed model was implemented and results of experiments are discussed.

Hoping that the above summarized collection of papers will be of interest to SCPE readers we would also like to invite everyone to the second edition of the IDC Symposium, which will take place on September 18–20, 2008, in Catania, Italy (http://idc08.diit.unict.it/).

Maria Ganzha and Marcin Paprzycki,
IBS PAN, Warszawa

Scalable Computing: Practice and Experience
Volume 9, Number 1, pp. 1–10. http://www.scpe.org

ISSN 1895-1767
© 2008 SCPE

STUDYING SVM METHOD'S SCALABILITY USING TEXT DOCUMENTS

DANIEL MORARIU*, MARIA VINŢAN†, AND LUCIAN VINŢAN*

Abstract. In the last years the quantity of text documents is increasing continually and automatic document classification is an important challenge. In the text document classification the training step is essential in obtaining good results. The quality of learning depends on the dimension of the training data. When working with huge learning data sets, problems regarding the training time that increases exponentially are occurring. In this paper we are presenting a method that allows working with huge data sets into the training step without increasing exponentially the training time and without significantly decreasing the classification accuracy.

Key words. text mining, classification, clustering, kernel methods, support vector machine

1. Introduction.

While more and more textual information is available online, effective retrieval is difficult without good indexing and summarization of documents' content. Documents' categorization is one solution to this problem. The task of documents' categorization is to assign a user defined categorical label to a given document. In recent years a growing number of categorization methods and machine learning techniques have been developed and applied in different contexts.

Documents are typically represented as vectors in a features space. Each word in the vocabulary is represented as a separate dimension. The number of word occurrences in a document represents the normalized value of the corresponding component in the document's vector.

In this paper we are investigating how classification accuracy is influenced using only relevant selected input vectors subset belonging to a larger data training set. The answer will show if the classification method is scalable or not. We are using classifiers based on Support Vector Machine (SVM) techniques. They are less vulnerable to degrade when the dimensionality of the feature space is increasing, and have been shown effective in many classification tasks. The SVM is actually based on learning with kernels and support vectors.

We are developing a seven steps strategy that trains the classifiers on a reduced data set without significant accuracy decrease comparing with training on the larger initial data set, but reducing the learning time [1]. In designing this strategy we were inspired by a method presented in [2] which uses a tree structure to hierarchically group similar databases articles, at different abstract levels. Despite this strategy was not recommended by the authors to be used on text documents, we modified it in order to group the similar text documents into a single level.

In this experiment we are using an original implementation of SVM algorithm with some improvements presented by us in [3]. In order to represent the vectors and to select the relevant features we are using a feature selection method based on SVM, already presented in [4, 5]. For training and testing part we are using Reuters' database [6].

Section 2 contains the prerequisites for the work that we are presenting in this paper. In sections 3 we are presenting the proposed scalability method and in section 4 the main results of our experiments. The last section debates and concludes on the most important obtained results and proposes some further work, too.

2. Experimental Framework.

2.1. Support Vector Machine.

The Support Vector Machine (SVM) is a classification technique based on statistical learning theory [7, 8] that was successfully applied in many challenging non-linear classification problems, processing large data sets.

The SVM algorithm finds a hyperplane that optimally splits the training set. The optimal hyperplane can be distinguished by the maximum margin of separation between all training points and itself [9]. Looking at a two-dimensional space we actually want to find a line that "best" separates points in the positive class from points in the negative class. The hyperplane is characterized by a decision function like:

$$f(x) = \operatorname{sign}(\langle \vec{w}, \Phi(x) \rangle + b) \tag{2.1}$$

* "Lucian Blaga" University of Sibiu, Computer Science Department, Emil Cioran street, no. 4, Sibiu 550025, Romania ({daniel.morariu, lucian.vintan}@ulbsibiu.ro).

† "Lucian Blaga" University of Sibiu, Electric and Electronic Engineering Department, Emil Cioran street, no. 4, Sibiu 550025, Romania (maria.vintan@ulbsibiu.ro).

where $\vec{w}$ is the weight vector, orthogonal to the hyperplane, "b" is a scalar that represents the margin of the hyperplane, "x" is the current tested sample, "$\Phi(x)$" is a function that transforms the input data into a higher dimensional feature space and $\langle \cdot, \cdot \rangle$ represents the dot product. Sign is the sign function. If $\vec{w}$ has unit length, then $\langle \vec{w}, \Phi(x) \rangle$ is the length of $\Phi(x)$ along the direction of $\vec{w}$. Generally $\vec{w}$ will be scaled by $\|\vec{w}\|$. In the training part the algorithm needs to find the normal vector "$\vec{w}$" that leads to the largest "b" of the hyperplane. For extending the SVM algorithm from two-class classification to multi-class classification typically one of two methods is used: "One versus the rest", where each topic is separated from the remaining topics, and "One versus the one", where a separate classifier is trained for each class pair. We selected the first method for two reasons: first, our preliminary experiments show that the first method obtains better performance, which might be explained by the fact that the Reuters' database contains strongly overlapped classes and assigns almost all samples in more than one class. Second, the overall training time is shorter for the first method.

2.2. The Data-set. Our experiments were performed on the Reuters-2000 collection [6], which has 984MB of newspapers articles in a compressed format. Collection includes a total of 806,791 documents, with news stories published by Reuters Press covering the period from 20.07.1996 through 19.07.1997. The articles have 9822391 paragraphs and contain 11522874 sentences and 310033 distinct root words. Documents are pre-classified according to 3 categories: by the Region (366 regions) the article refers to, by Industry Codes (870 industry codes) and by Topics proposed by Reuters (126 topics, 23 of them contain no articles). Due to the huge dimensionality of the database we will present here results obtained using a data subset. From all documents we selected the documents with the industry code value equal to "System software". We obtained 7083 files that are represented using 19038 features and 68 topics. We represent a document as a vector of words, applying a stop-word filter (from a standard set of 510 stop-words [10]) and extracting the word stem [10]. From these 68 topics we have eliminated those that are poorly or excessively represented, obtaining 24 different topics and 7053 documents that were split randomly in a training set (4702 samples) and a testing set (2351 samples).

In some experiments we will use a larger data set of 16139 documents. From entire Reuters Database we have selected those documents that are grouped by Reuters in "Computer Systems and Software" (I33020) by industry code. After this selection there are 16139 documents left to work on. From these documents we extracted a number of 28624 features after eliminating the stop-words and stemming. The documents are pre-classified by Reuters after topic and in the resulting set there were obtained 25 different topics. These 16139 samples are divided randomly into a training set of 10757 samples and a testing set of 5382 samples.

2.3. Kernel Types. The idea of the kernel trick is to compute the norm of the difference between two vectors in a higher dimensional feature space without representing them in that space. We are using in our selected classifiers two types of kernels, each of them with different parameters: polynomial and Gaussian kernels [3]. For the polynomial kernel we vary only the kernel's degree and for the Gaussian kernel we change the parameter C according to the following formulas (x and x' being the input vectors):

Polynomial,

$$k(x, x') = (2 \cdot d + \langle x \cdot x' \rangle)^d \tag{2.2}$$

where d being the only parameter to be modified and representing the kernel's degree:

Gaussian (radial basis function RBF),

$$k(x, x') = \exp(-\frac{\|x - x'\|^2}{n \cdot C}) \tag{2.3}$$

where C being the classical parameter and n being the new parameter, introduced by us, representing the number of elements from the input vectors that are greater than 0 [3].

2.4. Representing the input data. Also in our work we will use different representations of the input data. After extensive experiments [5],[11] we have seen that different types of kernels work better with different types of data representation. We represented the input data in three different formats. In the following formulas *n(d,t)* is the number of times that term t occurs in document d, and $\max_\tau n(d, \tau)$ is the maximum frequency occurring in document d [12].

Binary representation—in the input vector we store "0" if the word doesn't occur in the document and "1" if it occurs, without considering the number of occurrences.

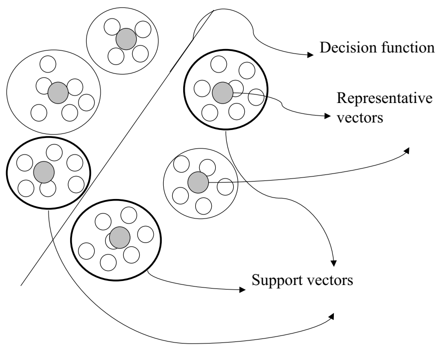


FIG. 3.1. *Selecting of support vectors.*

Nominal representation—we compute the value of the weight using the formula:

$$TF(d,t) = \frac{n(d,t)}{\max_\tau n(d,\tau)} \tag{2.4}$$

Cornell SMART representation—we compute the value of the weight using the formula:

$$TF(d,t) = \{ \begin{array}{cc} 0 & if\, n(d,t) = 0 \\ 1 + \log(1 + \log(n(d,t)) & otherwise \end{array} \tag{2.5}$$

3. A Method for studying the SVM Algorithm's Scalability. The original learning step that uses the Support Vector Machine technique to classify documents is split in three general stages. In the first stage all training data are grouped based on their similarity. For compute the similarity we use two different methods based on the "winner-take-all" idea [13]. The first method computes the center of the class (called further the representative vector) using arithmetic average and the second method computes the center of the class using LVQ (Learning Vector Quantization—equation (3.2)) in one step. The number of groups that can be created in this stage is unlimited and depends on the similarity level, data dimension and a specified threshold. From each created group a representative vector is obtained. With these vectors we create a new training data set that will be used in the classification step. After the classification (developed using Support Vector Machine [7]), besides the classification rules (decision functions) that are obtained, we also obtain the elements that have an effective contribution to the classification (named support vectors in the SVM algorithm). Taking only the support vectors (called further relevant vectors) we obtained a new reduced data set, containing only a small but relevant part of the representative vectors already obtained in the first stage. For each relevant vector we take all original vectors that were grouped in its category and generate a new data set. In the third stage, a new classification step, we will use a reduced dimension of the training set choosing only a small part of the input vectors that can have a real importance in defining the decision function. All these stages are represented in Fig. 3.1, too.

The original learning step (those three previously presented stages) that uses the Support Vector Machine technique to classify documents are splits in the next 7 smaller steps:

1. We normalize each input vector in order to have the sum of elements equal to 1, using the following formula:

$$\widetilde{TF}(d,t) = \frac{n(d,t)}{\sum_{\tau=1}^{19038} n(d,\tau)} \tag{3.1}$$

where $TF(d,t)$ is the term's frequency, $n(d,t)$ is the number of times that term t occurs in document d, and the denominator represents the sum of terms that occur in the entire document d.

2. After normalization we compute the Euclidian distance between each input vector and each center of the group (*representative vector*) that was created up to this moment(the gray small circles in Fig. 3.1). We keep the smallest obtained distance (linear clustering). If this distance is smaller than a predefined threshold we will introduce the current sample into the winner group and recompute the representative vector for that group; if not, we will create a new group and the current input vector will be the representative vector for that new group.

3. After grouping, we create a new training data set with all the representative vectors (all the gray circles in Fig. 3.1). This set will be used in the classification step. In this step we are not interested in the classification accuracy because the vectors are not the original vectors. Here, we are interested only in selecting relevant vectors (the vectors that have an effective contribution to the classification).

4. We are doing a feature selection step on this reduced set (each of them having 19038 features). For this step we are using SVM_FS method presented in [4]. After computing all weights we select 1309 features because, as we showed in [11], this number of features produces optimal results.

5. The resulted smaller vectors are used in a learning step. For this step we use polynomial kernel with degree equal to 1 and nominal data representation. We use the polynomial kernel because it usually obtains a smaller number of support vectors in comparison with the Gaussian kernel [11]. We use the kernel's degree equal to 1 because in almost all previous tests we obtained better results with this value.

6. After SVM learning step, besides the classification rules (decision functions) that are obtained, we also obtained the elements that have an effective contribution to the classification (named support vectors in the SVM algorithm). They are represented in Fig. 3.1 as being the large circles with a thick. We chose all groups that are represented by those selected vectors and we develop a new set containing only the vectors belonging to these groups. This new set is a reduced version of the original set, containing only *relevant input vectors* that are considered to have a significant influence on the decision function.

7. This reduced vectors set will now be used in the feature selection and classification steps as the original input data.

In the second presented step all training data are grouped based on their similarity. To compute the similarity we use two different methods based on the "winner-takes-it-all" idea [13]. First method computes the center of the class (the representative vector) using arithmetic mean and the second method computes the center of the class using LVQ formula (Learning Vector Quantization [14]) in one step. Depending on the method used to compute the representative vector we are using two vector representation types. In the first method, where the representative vector is computed as an arithmetic mean, it contains the sum of all elements that are included in that group, and a value that represents the total number of samples from that group. In the second method the representative vector is computed using equation (3.2) for each new sample that is included in the group. The formula for computing the representative vector for the LVQ method is:

$$\vec{w}_i(t+1) := \vec{w}_i(t) + \alpha(\vec{x} - \vec{w}_i(t)) \tag{3.2}$$

where $\vec{w}_i$ is the representative vector for the class i, $\vec{x}$ is the input vector and α is the learning rate. Because we want a one step learning and taking into account the small number of samples, we'll choose a relatively great value for the parameter α.

3.1. Clustering Data Set using Arithmetic Mean. In our presented results we started with an initial set of 7083 vectors. After the grouping step we reduce this dimension at 4474 representative vectors, meaning 63% of the initial set. For this reduction we used a threshold equal to 0.2. On this reduced set a classification step for selecting the relevant vectors was made. After the classification the algorithm returns a number of 874 support vectors. Taking those support vectors we created a data set containing only 4256 relevant samples meaning approximately 60% from the initial set. Using this new reduced set we developed a feature selection step, using SVM_FS method [4], and we select only 1309 relevant features. The reduced set is split in a training set having 2555 samples and in a testing set having 1701 samples.

3.2. Clustering Data Set using the LVQ Algorithm. In order to compare the two presented methods we tried to obtain approximately the same number of vectors with both methods. With the LVQ algorithm, for a value of the threshold equal to 0.15 and a learning rate equal to 0.9, we obtained after the first step 3487 groups (representative vectors). We expect this method to work better when huge data sets will be used. In the next step, we trained a classifier in order to select from these representative vectors only those vectors that are relevant. The algorithm returns 912 support vectors and we selected only those support vectors that have the

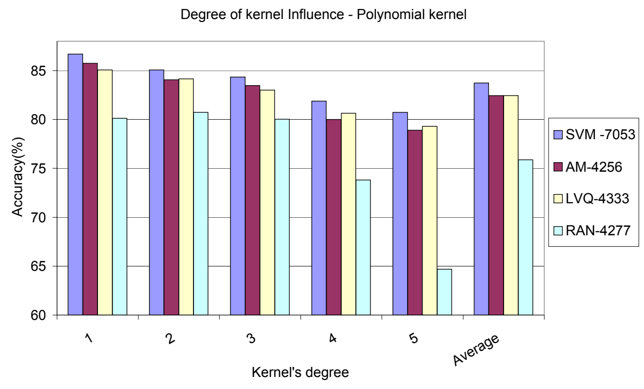


FIG. 4.1. *Comparative results for different data set dimensions—Polynomial kernel.*

Lagrange multipliers greater than a certain threshold (in our case 0.25). We considered these support vectors as being the most relevant vectors from all the representative vectors. We create a data set that contains only 4333 samples. This number represents approximately 61% from the initial data set. The obtained set is split randomly into a training set of 2347 samples and in a testing set of 1959 samples.

4. Experimental Results.

4.1. Methods'Scalability—Quantitative Aspects. We are presenting here results only for a reduced vector dimension as number of features (1309 features). We are using this dimension because with this number of features we obtained the best results [5]. Also in our work we are using three different representations of the input data: binary, nominal and Cornell Smart [3].

In Fig. 4.1 we are presenting comparative classification accuracies obtained for Polynomial kernel and nominal data representation for all four sets—the original set noted as SVM-7053, the set obtained using arithmetic mean to compute the representative vector, noted as AM-4256, the set obtained using the LVQ method to compute the representative vector, noted as LVQ-4333 and the set randomly obtained, noted as RAN-4277. The RAN-4277 set is obtained choosing randomly a specified number of samples from the original set [11].

As it can be observed there is a small difference between the results obtained for AM-4256 and LVQ-4333. The difference in the accuracy obtained between the original set and AM-4256 set is on average equal to 1.30% for all kernel degrees. The same average difference is obtained also between the original set and LVQ-4333. When we work with a small degree of the kernel the difference between the original set and AM-4256 set is smaller than 1% but the difference between the original set and LVQ-4333 is greater (1.60%). When the kernel's degree increases the results are better with LVQ-4333 comparatively with AM-4256 but usually the difference can be considered insignificant. At average, over all kernels' degree and all data representations the difference between the original set and AM-4256 is 1.65% and the difference between the original set and LVQ-4333 is 1.64%. We observe that for the same values of kernel's degree, which obtained the best accuracy using the original set, were also obtained the smallest difference between the original set and the reduced set. The results obtained using randomly choused set are at average with 8% smaller than the results obtained using the entire set.

In Fig. 4.1 it is interesting to observe that for small kernel's degree we obtain better results using first method for creating the groups and for the greater kernel degrees we obtain better results using second method for creating the groups. In the theory of Support Vector Machine researchers recommend to use greater value for the degree in case where data are overlapped. In the case when data are not strongly overlapped, with small

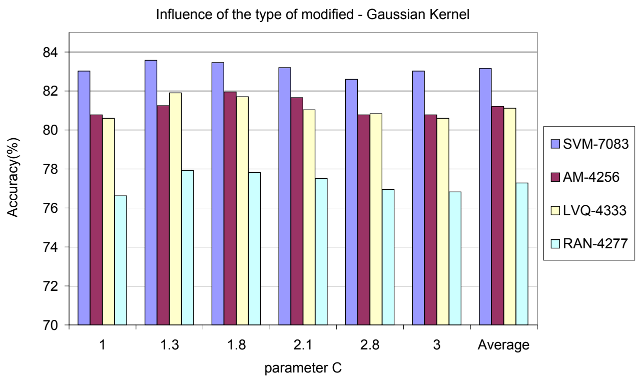


Fig. 4.2. *Comparative results for different data set dimensions—Gaussian kernel.*

degrees are obtained better results and with greater degrees the accuracy of classification decreases. Obtained results suggest that with the first method usually we haven't obtained overlapped data but with the second methods based on LVQ we obtained overlapped data. This observation occurs also for Binary and Cornell Smart data representation and Polynomial kernel. An interesting behavior can be observed when the kernel's degree increases (and the accuracy of classification decreases) the discrepancy between the original set and both reduced sets decreases, too. We developed also some tests with different values for threshold and learning rate (α) and this value only increases more or less the number of groups. Usually for both methods the threshold is the main parameter that adjusts the number of classes. The parameter (α) has only a small influence in determining the number of classes (depending the input set dimension and on the on-coming samples), but has a great influence in specifying the center of the class. Parameter (α) in this tested case has a small influence because we worked with a relatively reduced number of samples. Even if the LVQ-4333 uses more samples for training in the second step comparatively with AM-4256 the results are usually poor because in the first step the representative vectors aren't able to represent exactly the center of the groups; the selection of the representative vectors maybe can have a great influence in defining the hyperplane even if not all vectors will be relevant in the final step.

In Fig. 4.2 are presented the results obtained using the Gaussian kernel and Cornell Smart data representation. For this kernel the average accuracy difference between the two sets is greater than for the polynomial kernel case, being at average of 1.89% for AM-4256 and 1.95% for LVQ-4333. The difference between the original set and the randomly reduced set (RAN-4277) is also big for the Gaussian kernel (being at average 7%). The smallest difference was obtained with a parameter C equal with 1.8. For this value we obtained the best results in all previous tests (using the original data set). This difference is of 1.5% for AM-4256 and of 1.75% for LVQ-4333. For this type of kernel the method based on LVQ obtains poorly results in almost all cases.

We are reducing the data in the first step at 63% and in the second step at 60% for the first method and respectively to 50% in the first step and 61% in the second step for the LVQ method. With this reduction however the lose in accuracy was about 1% for polynomial kernel and about 1.8% for Gaussian kernel. It is interesting to note that the optimal parameter's values (degree or C) are usually the same for the original data set and respectively the reduced one.

Obviously, the time needed for training on a smaller number of samples decreases. For example, for polynomial kernel with degree 1 and Nominal data representation, 1031 seconds are needed to learn using all data set and 209 seconds using the reduced set. At these 209 seconds we also have to add the time needed to select the support vectors (548 seconds) and the time needed for grouping data (84 seconds). The last two times occur only once for all the tests with polynomial and Gaussian kernels. The total time for polynomial kernel

TABLE 4.1
Training time for each data set and some training characteristic.

Data set	Kernel's Characteristics	t_{group_time}	t_{select_SV}	$t_{classify}$	$t_{training\_total}$
SVM-7083	POL, D2.0, BIN	-	-	-	1532.57
SVM-7083	POL, D1.0, NOM	-	-	-	1031.21
SVM-7083	POL, D1.0, CS	-	-	-	1107.64
SVM-7083	RBF, C2.8, BIN	-	-	-	4492.95
SVM-7083	RBF, C3.0, CS	-	-	-	4481.65
AM-4256	POL, D2.0, BIN	84	548.46	263.36	895.82
AM-4256	POL, D1.0, NOM	84	548.46	209.62	842.08
AM-4256	POL, D1.0, CS	84	548.46	215.56	848.02
AM-4256	RBF, C2.8, BIN	84	548.46	511.87	1144.33
AM-4256	RBF, C3.0, CS	84	548.46	513.81	1146.27
LVQ-4333	POL, D2.0, BIN	97	571.43	289.40	957.83
LVQ-4333	POL, D1.0, NOM	97	571.43	232.53	900.96
LVQ-4333	POL, D1.0, CS	97	571.43	250.67	919.10
LVQ-4333	RBF, C2.8, BIN	97	571.43	599.14	1267.57
LVQ-4333	RBF, C3.0, CS	97	571.43	618.04	1286.47

TABLE 4.2
Decreasing in average accuracy for all data representation.

Kernel type	Data representation	Average accuracy[%] AM-4256	Average accuracy[%] LVQ-4333	Average accuracy[%] Ran-4277
Polynomial kernel	BINARY	1.907259	2.124463	7.581452
Polynomial kernel	NOMINAL	1.302935	1.307764	7.869350
Polynomial kernel	CORNEL SMART	1.750215	1.943873	7.603154
Gaussian kernel	BINARY	3.816914	3.755501	8.776014
Gaussian kernel	CORNEL SMART	1.891519	1.954537	5.870109

and degree 1 is 842 seconds. To compute these timing, in both cases (with the original set and with the reduced set), we don't take into consideration the time needed for feature selection. Every time the feature selection step starts with 19038 features but in the second case we have a reduced dimension of the set (as number of vectors). Some of these timing are also smaller than the first time. These values were obtained using a Pentium IV at 3.2 GHz, 1GB DRAM memory and Win XP. For the second method to obtain the representative vectors (using LVQ) the grouping part takes more time (97 seconds). The time for selecting support vector is 571 seconds. For example the time needed to compute the classification accuracy for polynomial kernel and degree 2 is 232 seconds, so the total time that can be considered is computed as:

$$t_{total\_time} = t_{group\_time} + t_{select\_SV} + t_{classify} \tag{4.1}$$

where t_{group_time} is the time needed for grouping data, t_{select_SV} is the time needed for classifying data and finding support vectors and $t_{classify}$ is the time needed for classifying the reduced set. This time can also include the time needed for selecting the features. This time is not included in the classifying time using original data set so it will not be included here.

In Table 4.1 are presented some training times for each data set. We are interested only in training time because after training the testing time depends only on the testing set dimension and the number of support vectors. For only one sample the response is less than one second.

In Table 4.2 the average difference between the accuracy obtained with the original set and the accuracy obtained with the reduced set are presented. In other words, we present here the average decrease obtained for each type of data representation and for each kernel when we worked with a reduced set. For Gaussian kernel and Binary data representation we obtained the greatest decrease from all the tests (3.8%). For polynomial kernel the decrease is at average of 1.65%.

Learning with the entire Reuters' data set will be impossible on ordinary PC's due to the huge time and memory needed. Our obtained results will be useful in choosing the adequate parameters for the learning step. For example, for the entire Reuters database we only started the first step of feature extraction and after this step we obtained 806791 vectors, each of them having 310033 dimensions (features) and a total of 103 distinct topics.

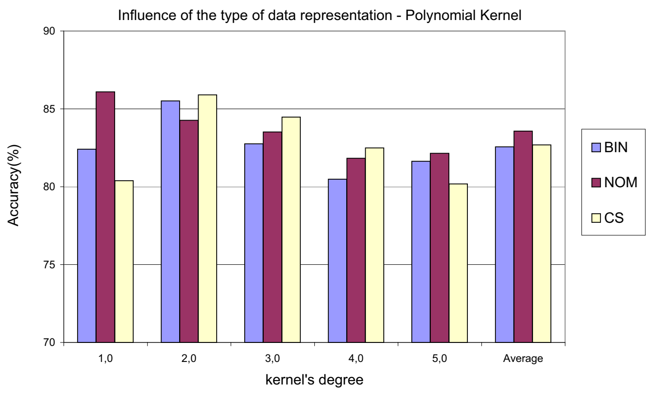


FIG. 4.3. *Influence of kernel's degree using a large data set—Polynomial kernel.*

4.2. Expected Results using a Larger Data set. We chose a significantly larger training and testing data sets in order to test the previously presented method. We can not run tests for this larger data set due to its dimensionality and the limits of our computational capability but we reduced it and perform tests only for the reduced dimension. Based on the obtained results presented in the previous section we will try to estimate the performance that can be obtained if the entire set will be used. The characteristics for this large data set were presented in section 2.2. First, we tried to create small groups in order to reduce the dimension following the steps presented in section 3 and depicted in Fig 3.1. For creating the groups we use the method based on the LVQ algorithm executed in just one step. Thus, using a threshold equal with 0.15 and a learning coefficient α=0.4 (smaller because a large number of vectors are used) after a first step we obtained 11258 groups that represent a reduction at 69% of the entire set. Using this reduced set we start training using SVM algorithm in order to select only the support vectors. After training we select only those support vectors that have values greater than a threshold equal with 0.2 (only relevant vectors). After this second step we obtained only 10952 samples that are split randomly in a training set of 5982 samples and a testing set of 4970 samples. In what follows we present the results obtained using only this reduced set that means 67% of the entire set. If the scalability is kept according to our figures presented in the previous paragraph, where the results are usually with 1-2% smaller if the reduced set is used instead of the entire set, we will expect that, if the entire large set will be used, the accuracy will be with only 1-2% higher than the presented results.

We run a lot of tests for polynomial kernel and a lot of tests for Gaussian kernel using all of the three types of data representation. We present results comparatively for each type of data representation. Fig. 4.3 presents results obtained for polynomial kernel and all types of data representation. The presented results are obtained using a set with 3000 relevant features. For features selection we used SVM_FS method that was presented in [5]. Even if we work on the reduced large set, the results are comparable with results obtained working with the entire set but with a smaller number of samples (7053 samples). For example, for polynomial kernel and nominal data representation using a set of 7053 samples we obtained an average accuracy of 83.73% and using this reduced data set we obtained 83.57% at average.

Fig. 4.4 presents results obtained using Gaussian kernel and several values for parameter C using two types of data representation. For Gaussian kernel in average with this new data we obtained for Cornell Smart data representation an average accuracy of 82.57% which is closer to the average accuracy obtained with a set with 7053 samples 83.14%. All the presented results are obtained using a reduced set at 65% of the entire data set. We will not present here the time for training and testing because we used on the training part only the reduced set and we don't have results for the training time of the entire set.

5. Conclusions and Further Work. In the real life when working with text documents we need to work with huge sets of documents to obtain good classification accuracy. In this paper we are proposing and

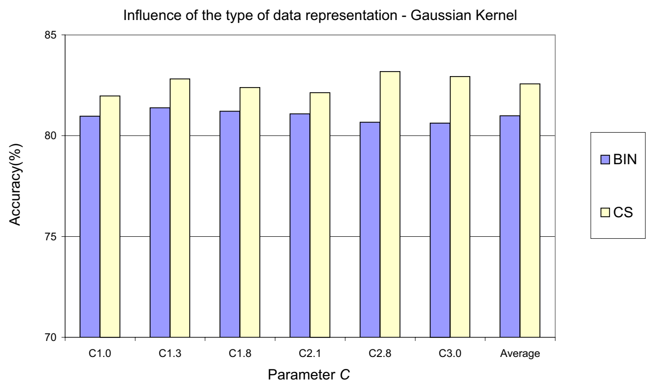


FIG. 4.4. *Influence of C parameter using a large data set—Gaussian kernel.*

developing a strategy to work with large text documents sets. This strategy doesn't increase exponentially the training time, actually it decreases it and doesn't substantially loose in classification accuracy. A method to reduce the number of vectors from the input set and make two learning steps in order to consider the learning step finished was developed and tested. We noticed that the classification accuracy decreases at average with only 1% for polynomial kernel and with about 1.8% for Gaussian kernel, when the data set is reduced at 60% of the entire data set. If the reduction of the set was randomly made (RAN-4277) at a dimension smaller with 40% than the original set, the lost in classification accuracy is at average of 8% for the polynomial kernel and respectively of 7% for the Gaussian kernel.

A major issue that occurs in all classification and clustering algorithms is that they are reluctant to fit in the real spaces. For instance they have a problem dealing with new documents for which none of the features are in the previous feature set (the product between the new features set and the previous feature set is an empty set). As a further improvement we will try to develop tests with families of words and use as features only a representative of each family. In this way the number of features will be significantly reduced and thus we can increase the number of files that can be classified further on. In order to achieve this we might use the WordNet database, which contains a part of the families of words for the English language.

Acknowledgments. We would like to thank to SIEMENS AG, CT IC MUNCHEN, Germany, especially to Vice-President Prof. Hartmut RAFFLER and to Dr. Volker Tresp, for their generous and various supports that they have provided in developing this work.

REFERENCES

[1] D. MORARIU, M. VINTAN AND L. VINTAN, *Aspects Concerning SVM Method's Scalability*, Proceedings of the 1st International Symposium on Intelligent and Distributed Computing—IDC2007, ISBN 978-3-540-74929-5, ISSN 1860-949x, Craiova, October, 2007.

[2] H. YU, J. YANG AND J. HAN, *Classifying Large Data Sets Using SVM with Hierarchical Clusters*, in SIGKDD03 Exploration: Newsletter on the Special Interest Group on Knowledge Discovery and Data Mining, ACM Press, Washington, DC, USA, 2003.

[3] D. MORARIU AND L. VINTAN, *A Better Correlation of the SVM Kernel's Parameters*, Proceedings of the 5th RoEduNet IEEE International Conference, ISBN (13) 978-973-739-277-0, Sibiu, June, 2006.

[4] D. MORARIU, L. VINTAN AND V. TRESP, *Feature Selection Method for an Improved SVM Classifier*, Proceedings of the 3rd International Conference of Intelligent Systems (ICIS'06), ISSN 1503-5313, vol. 14, pp. 83-89, Prague, August, 2006.

[5] D. MORARIU, L. VINTAN AND V. TRESP, *Evaluating some Feature Selection Methods for an Improved SVM Classifier*, International Journal of Intelligent Technology, Volume 1, no. 4, ISSN 1305-6417, pages 288-298, December, 2006.

[6] M. WOLF AND C. WICKSTEED, *Reuters Corpus:* `http://www.reuters.com/researchandstandards/corpus/` Released in November 2000 accessed in June 2005.

[7] B. SCHOLKOPF AND A. SMOLA, *Learning with Kernels, Support Vector Machine*, MIT Press, London, 2002.

[8] C. Nello and J. Shawe-Taylor, *An Introduction to Support Vector Machines and other kernel-based learning methods*, Cambridge University Press, 2000.
[9] V. Vapnik, *The nature of Statistical learning Theory*, in Springer New York, 1995.
[10] `http://www.cs.utexas.edu/users/mooney/ir-courses/`—Information Retrieval Java Application.
[11] D. Morariu, *Relevant Characteristics Extraction*, in 3rd PhD Report, University "Lucian Blaga" of Sibiu, October, 2006, `http://webspace.ulbsibiu.ro/daniel.morariu /html/Docs/Report3.pdf`
[12] S. Chakrabarti, *Mining the Web. Discovering Knowledge from Hypertext Data*, Morgan Kaufmann Publishers, USA, 2003.
[13] C. Hung, S. Wermter and P. Smith, *Hybrid Neural Document Clustering Using Guided Self-Organization and WordNet*, in IEEE Computer Society, 2003.
[14] T. Kohonen, *Self-Organizing Maps*, Second edition, Springer Publishers, 1997.

Edited by: Maria Ganzha, Marcin Paprzycki
Received: Jan 21, 2008
Accepted: Feb 9, 2008

SCALABLE COMPUTING: PRACTICE AND EXPERIENCE
Volume 9, Number 1, pp. 11–20. http://www.scpe.org

ISSN 1895-1767
© 2008 SCPE

LEARNING OBJECTS' ARCHITECTURE AND INDEXING IN WELSA ADAPTIVE EDUCATIONAL SYSTEM[*]

ELVIRA POPESCU[‡,§,†], COSTIN BĂDICĂ[‡], AND PHILIPPE TRIGANO[§]

Abstract. In this paper we present an intelligent way of organizing learning material in an adaptive educational hypermedia system. We describe the use of instructional metadata which facilitates both the detection of student learning style and the application of various adaptation techniques. The advantage of our approach is that it is independent of a particular learning style model. Furthermore, the author has to supply only the annotated learning content (the static description) while the adaptation logic (the dynamic description) is provided by the system. The approach is implemented in an adaptive educational system called WELSA and illustrated with a course module in the area of Artificial Intelligence.

Key words. learning object, educational metadata, adaptive educational hypermedia, learner modelling, learning style

1. Introduction. Educational metadata is a special kind of metadata that provides information about learning objects. A learning object represents any reproducible and addressable digital resource that can be reused to support learning [20]. Currently there are several initiatives for standardizing educational metadata, addressing the issues of reusability, interoperability, discoverability, sharing and personalization [4].

IEEE LOM (Learning Object Metadata) [19] is the most prominent standard, being elaborated by the IEEE Learning Technology Standards Committee. IMS Global Learning Consortium also contributed to the drafting of the IEEE LOM and consequently the current version of IMS Learning Resource Metadata specification (IMS LRM v.1.3 [19]) is based on the IEEE LOM data model. LOM contains nine categories of metadata: General, Lifecycle, Meta-metadata, Technical, Educational, Rights, Relation, Annotation and Classification. The attributes that are relevant from the point of view of instruction and pedagogy are those pertaining to the Educational category, particularly the *Learning Resource Type.* Its possible values are: *Exercise, Simulation, Questionnaire, Diagram, Figure, Graph, Index, Slide, Table, Narrative Text, Exam, Experiment, Problem Statement, Self Assessment, Lecture.*

Another widely known standard is *SCORM (Sharable Content Object Reference Model)* [2] which originates from e-learning requirements of the US Armed Forces, being produced by ADLNet (Advanced Distributed Learning Network) initiative. SCORM includes three types of learning content metadata: raw media metadata (that provide information about assets independently of learning content), content metadata (that provide information about learning contents, independently of a particular content aggregation) and course metadata (that provide information about the content aggregation).

Dublin Core metadata standard [12] is a simple yet effective general-purpose metadata scheme, for describing a wide range of networked resources. It was developed within the Dublin Core Metadata Initiative (DCMI). At present, there is a joint DCMI/IEEE LTSC Task Force activity, with the objective of developing a representation of the metadata elements of the IEEE LOM in the Dublin Core Abstract Model.

The main problem with these specifications is that they fail to include the instructional perspective [29]. In case of LOM, the property *Learning Resource Type* attempts to address this issue, but mixes instructional and technical information. Thus some of the values describe the instructional role of the resource (*Exercise, Simulation, Experiment*), while others are concerned with their format (*Diagram, Figure, Graph, Slide, Table*). Moreover, some important instructional types are missing, such as *Definition, Example* or *Theorem.* In order to overcome this issue, Ullrich introduced an instructional ontology, which is domain independent and pedagogically sound [29]. One of the most important advantages of this ontology is its pedagogical flexibility, being independent of a particular instructional theory. Moreover, as we will show in section 3, the ontology can also be enhanced to serve adaptivity purposes, from the point of view of various learning styles.

The rest of the paper is structured as follows: the next section gives a short overview of adaptive educational systems that focus on the learning style of the students and sketches our approach. Section 3 describes the

[*]This is an extended version of the paper: Elvira Popescu, Costin Bădică, Philippe Trigano: Description and Organization of Instructional Resources in an Adaptive Educational System Focused on Learning Styles. In: *Advances in Intelligent and Distributed Computing, Proc. IDC 2007*, Craiova, Romania, *Studies in Computational Intelligence*, 78, 177–186, Springer, 2008.

[†]This author was supported by the Romanian Ministry of Education and Research, through CNCSIS grant TD 169/2007.

[‡]Software Engineering Department, University of Craiova, Romania ({popescu_elvira, badica_costin}@software.ucv.ro).

[§]Heudiasyc, UMR CNRS 6599, Université de Technologie de Compiègne, France (ptrigano@hds.utc.fr).

suggested organization of the learning resources and introduces the educational metadata that we propose to be used. Sections 4 and 5 illustrate the use of these metadata for learner modelling and adaptation provisioning respectively. Section 6 briefly introduces WELSA, an adaptive educational hypermedia system based on the above approach and describes a course module in the area of Artificial Intelligence, that was created and deployed using WELSA. Finally, some related works are presented in section 7 and conclusions are drawn in section 8.

2. Learning Style-based Adaptive Educational Systems. One of the most important goals of today's research in e-learning refers to the provision of an adaptive educational experience, that is individualized to the particular needs of the learner, from the point of view of knowledge level, goals or motivation. The purpose of this adaptation is to maximize the subjective learner satisfaction, the learning speed (efficiency) and the assessment results (effectiveness) [3].

Learning style-based adaptive educational systems (LSAES) are a special case of adaptive educational systems (AES), which focus on students' learning preferences as the adaptation criterion. According to [21], learning styles represent a combination of cognitive, affective and other psychological characteristics that serve as relatively stable indicators of the way a learner perceives, interacts with and responds to the learning environment. At present there is a large number of learning style models proposed in the literature (over 70 according to [10]), which differ in the learning theories they are based on, the number and the description of the dimensions they include. There are also a few educational systems that deal with them [27]. Some examples include: INSPIRE [23] (based on Honey and Mumford learning style model [18]), EDUCE [22] (based on Gardner's theory of multiple intelligences [15]), CS383 [8], Heritage Alive Learning System [9] and ILASH [5] (all based on Felder-Silverman learning style model [14]).

The main problem of the above systems is that they only take into account a single learning style model. Moreover, most of them use an explicit learner modelling method, asking the student to fill in a specialized psychological questionnaire. The resulted membership to a particular learning style is then stored once and for all in the student model kept by the system and it is subsequently used for adaptation. A few systems also adopt an implicit learner modelling method, trying to dynamically identify the student learning preferences by monitoring and analyzing student behaviour while it is using the system. The approach that we propose in [26] belongs to the latter category; furthermore it is not tied to a particular learning style model, but it integrates the most relevant characteristics from several models proposed in the literature, such as:

- perception modality (visual vs. verbal)
- processing information (abstract concepts and generalizations vs. concrete, practical examples; serial vs. holistic; active experimentation vs. reflective observation; careful vs. not careful with details)
- reasoning (deductive vs. inductive)
- organizing information (synthesis vs. analysis)
- motivation (intrinsic vs. extrinsic; deep vs. surface vs. strategic vs. resistant approach)
- pacing (concentrate on one task at a time vs. alternate tasks and subjects)
- social aspects (individual work vs. team work; introversion vs. extraversion; competitive vs. collaborative).

Our first objective is to dynamically model the learner, i. e. to identify the learning preferences by analyzing the behavioural indicators and then, based on them, infer the belonging to a particular learning style dimension. The second objective is to consequently adapt the navigation and the educational resources to match the student learning preferences (see figure 2.1 for a schematic description of the process). In order to achieve these two objectives, we need an intelligent way of organizing the learning material as well as a set of instructional metadata to support both learner modelling and adaptation processes.

3. Organizing the Educational Material in an LSAES. According to [20], learning objects represent any digital resources that can be reused to support learning. In our case, the most complex learning object (with the coarsest granularity) is the course, while the finest granularity learning object is the elementary educational resource. We have conceptualized the learning material using the hierarchical organization illustrated in figure 3.1: each course consists of several chapters, and each chapter can contain several sections and subsections. The lowest level subsection contains the actual educational resources. Each elementary learning object corresponds to a physical file and has a metadata file associated to it.

Based on our teaching experience, this is the natural and most common way a teacher is usually organizing his or her teaching materials. Additionally, this hierarchical approach presents several advantages, facilitating:

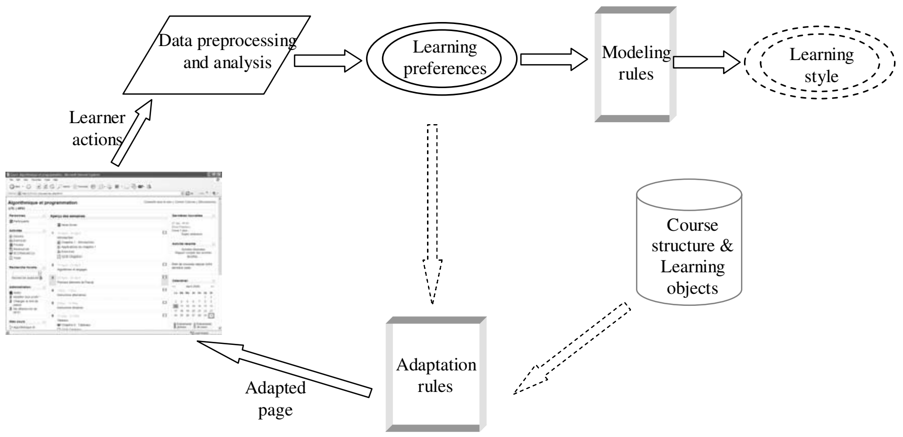


FIG. 2.1. *Schematic representation of our LSAES*

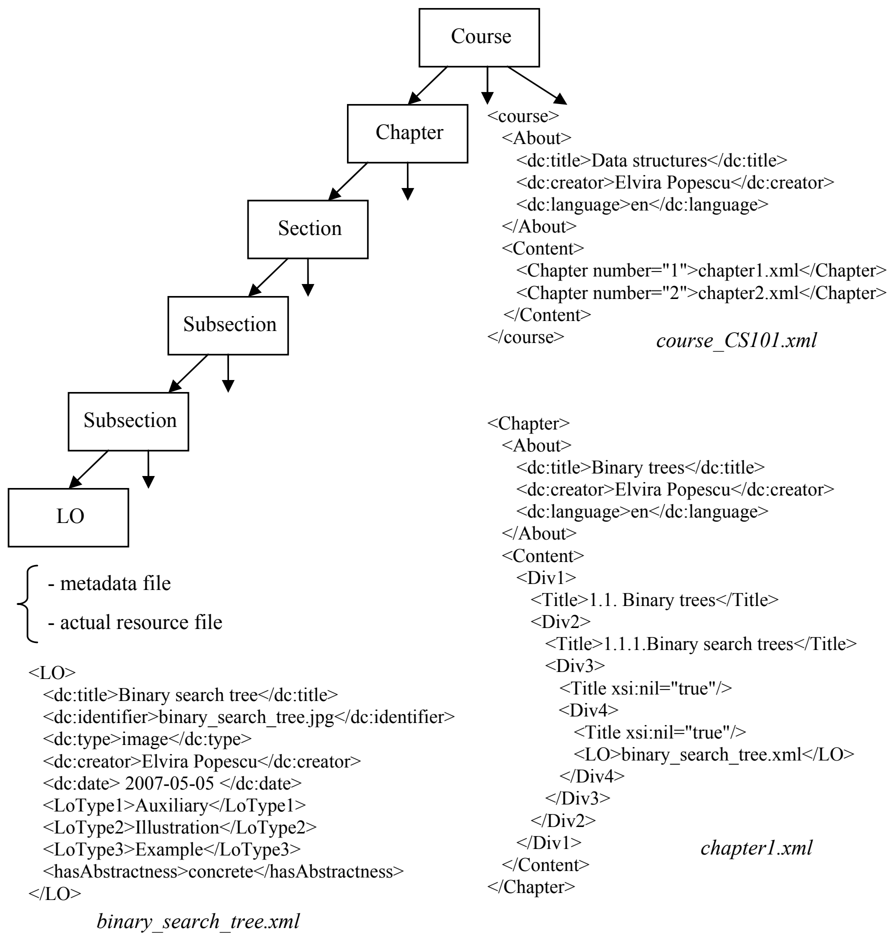
- high degree of reuse of the educational resources
- detailed learner tracking (since we are able to acquire and use all the information about what and how learning resources are accessed by which learners at a particular moment)—see section 4
- fine granularity of adaptation actions—see section 5.

As far as the educational metadata is concerned, one possible approach (which is used in [16]) would be to associate to each learning object the learning style that it is most suitable for. One of the disadvantages of this approach is that it is tightly related to a particular learning style. Moreover, the teacher must create different learning objects for each learning style dimension and label them as such. This implies an increase in the workload of the teacher, and also the necessity that she/he possesses knowledge of the learning style theory. Furthermore, this approach does not support dynamic learner modelling, since accessing a learning object does not offer sufficient information regarding the student (a learning object can be associated with several learning styles).

Instead, we propose a set of metadata that describe the learning object from several points of view, including: instructional role, media type, level of abstractness and formality, type of competence etc. These metadata were created by enhancing core parts of Dublin Core [12] and Ullrich's instructional ontology [29] with some specific extensions to meet the requirements of an LSAES. For example, some of the descriptors of a learning object that we propose are:

- *title* (the name given to the resource) → dc:title
- *identifier* (a reference to the actual resource, such as its URL) → dc:identifier
- *type* (the nature of the content of the resource, such as text, image, animation, sound, video) → dc:type
- *format* (the physical or digital manifestation of the resource, such as the media-type or dimensions of the resource) → dc:format
- *instructional role* that can be either i) *fundamental*: definition, fact, law (law of nature, theorem) and process (policy, procedure) or ii) *auxiliary*: evidence (demonstration, proof), explanation (introduction, conclusion, remark, synthesis, objectives, additional information), illustration (example, counter example, case study) and interactivity (exercise, exploration, invitation, real-world problem) → LoType1, LoType2, LoType3, LoType4.

Obviously, these descriptors are independent of any learning style. However, by analyzing the interaction of the student with the learning objects described by these metadata (for example by dynamically recording the time spent on each learning object, the order of access, the frequency of accesses), the system can infer a particular learning preference of the student. Furthermore, the teacher has to supply only annotated learning content (the static description) while the adaptation logic (the dynamic description) is provided by the system. This means that the adaptation rules are independent of the learning content and that they can be supplied by specialists in educational psychology. The next two sections illustrate our proposal of using these metadata for modelling the learner and for providing adaptation respectively.



```
<course>
  <About>
    <dc:title>Data structures</dc:title>
    <dc:creator>Elvira Popescu</dc:creator>
    <dc:language>en</dc:language>
  </About>
  <Content>
    <Chapter number="1">chapter1.xml</Chapter>
    <Chapter number="2">chapter2.xml</Chapter>
  </Content>
</course>
```

course_CS101.xml

```
<Chapter>
  <About>
    <dc:title>Binary trees</dc:title>
    <dc:creator>Elvira Popescu</dc:creator>
    <dc:language>en</dc:language>
  </About>
  <Content>
    <Div1>
      <Title>1.1. Binary trees</Title>
      <Div2>
        <Title>1.1.1.Binary search trees</Title>
        <Div3>
          <Title xsi:nil="true"/>
          <Div4>
            <Title xsi:nil="true"/>
            <LO>binary_search_tree.xml</LO>
          </Div4>
        </Div3>
      </Div2>
    </Div1>
  </Content>
</Chapter>
```

chapter1.xml

```
<LO>
  <dc:title>Binary search tree</dc:title>
  <dc:identifier>binary_search_tree.jpg</dc:identifier>
  <dc:type>image</dc:type>
  <dc:creator>Elvira Popescu</dc:creator>
  <dc:date> 2007-05-05 </dc:date>
  <LoType1>Auxiliary</LoType1>
  <LoType2>Illustration</LoType2>
  <LoType3>Example</LoType3>
  <hasAbstractness>concrete</hasAbstractness>
</LO>
```

binary_search_tree.xml

FIG. 3.1. *Suggested organization of the learning content in an LSAES*

4. Educational Metadata and Learner Modelling. The first step towards dynamic learner modelling comprises tracking and monitoring of student interactions with the system. Student observable behaviour in an educational hypermedia system includes: i) navigational indicators (number of hits on educational resources, navigation pattern); ii) temporal indicators (time spent on different types of educational resources proposed); iii) performance indicators (total learner attempts on exercises, assessment tests). Based on the interpretation of these observable facts, the system can infer different learning preferences. Knowing of the media type, the instructional role as well as other characteristics of the learning object the student interacts with is essential for an accurate identification of the learning preferences. Figure 4.1 illustrates the possible use of some of the learning object metadata.

5. Educational Metadata and Adaptation Logic. In the context of our work, modelling the learner is not a goal in itself.

The value of possessing a student model lies in its usability for providing a learning experience which is most beneficial for the student. Specifically, this could mean several things: in some cases, the most suitable attitude is to offer to the student the educational resources that better match his/her learning preferences, in terms of media type, browsing order of resources, communication and collaboration facilities, level of navigation guidance etc. In other situations, students could benefit more from being faced with a mismatched learning environment, which provides the necessary challenge to boost learning [22]. Moreover, when learners are firstly offered an educational content that doesn't match well their learning preferences, they will usually not limit themselves to that particular resource, being inclined to access more of the available resources on the subject.

The application of one or the other of the above methods depends on the intended pedagogical objective and on the characteristics of the target students (knowledge level, motivation, goals). The advantage of our

Learning preference	Behavioral indicators	Corresponding metadata tag
Visual preference	High amount of time spent on contents with graphics, images, video	<dc:type>image</dc:type> <dc:type>video</dc:type>
Verbal preference	High amount of time spent on reading text	<dc:type>text</dc:type> <dc:type>audio</dc:type>
Abstract concepts and generalizations	Access of abstract content first (concepts, definitions) High amount of time spent on abstract content	<LoType2>Definition</LOType2> <LoType2>Law</LOType2> <hasAbstractness>abstract</hasAbstractness>
Concrete, practical examples	Access of concrete content first (examples) High amount of time spent on concrete content	<LoType2>Illustration</LoType2> <LoType2>Fact</LOType2> <hasAbstractness>concrete</hasAbstractness>
Active experimentation	Access of practical content (simulations, exercises, problems…) before theory	<LoType2>Interactivity</LoType2>
Reflective observation	Access of theoretical content before practical content	<LoType1>Fundamental</LoType1>
Synthetic	High performance on exercises requiring synthesis competency	<hasCompetency>synthesis</hasCompetency>
Analytic	High performance on exercises requiring analysis competency	<hasCompetency>analysis</hasCompetency>

FIG. 4.1. *Correspondence between learning preferences and educational metadata*

approach is that it allows complete independence between the learner model and the pedagogical model: various adaptation actions can be associated with each learner preference. Furthermore, we could combine several pedagogical goals, using some of the identified learning preferences to improve the efficiency of the learning process (matching), others to provide the needed challenge and variety or to develop weaker skills (mismatching) and others to increase student's self-awareness about her/his strengths and weaknesses in the learning process (open model approach). Figure 5.1 illustrates a possible use of the detected learning preferences for a particular student. The adaptation techniques suggested are classified according to the levels of adaptation identified in [7] and [3].

Learning preference	Matched learning experience	Adaptation techniques
Visual	The course should include plenty of multimedia objects based on video and images; the content will be presented as much as possible using graphics and schemas.	Content level adaptation (specific media type filtering)
Concrete, practical examples	The course should be focused more on facts, practical aspects and examples. Each new concept will be first illustrated by an example and only then the theoretical aspects will be covered.	Content level adaptation (content hiding, specific item filtering) Presentation level adaptation (sorting fragments, dimming fragments)
Holistic	The course will include outlines and summaries for each course item, which will be presented at the beginning and end of each chapter and will be permanently accessible through a menu. The links to related or complex topics will be integrated in the content, to help situate the learnt subject and contribute to create the big picture. The exercises will be placed at the end of the chapter, not after each course item, in order to give the users the opportunity to holistically understand the subject first	Navigation level adaptation (link annotation, link generation) Content level adaptation (additional explanations) Presentation level adaptation (inserting fragments, sorting fragments)

FIG. 5.1. *Ways of providing adaptivity for different learning preferences*

As we can see, the adaptation techniques can be decomposed into elementary adaptation actions (sorting/inserting/removing learning objects) based on various criteria, all of which are included in the metadata:

- media type → dc:type
- instructional role → LoType1, LoType2, LoType3, LoType4
- level of abstractness → hasAbstractness
- type of competency required (in case of exercises) → hasCompetency

A formal representation of the adaptation knowledge as sets of rules is discussed in [25]

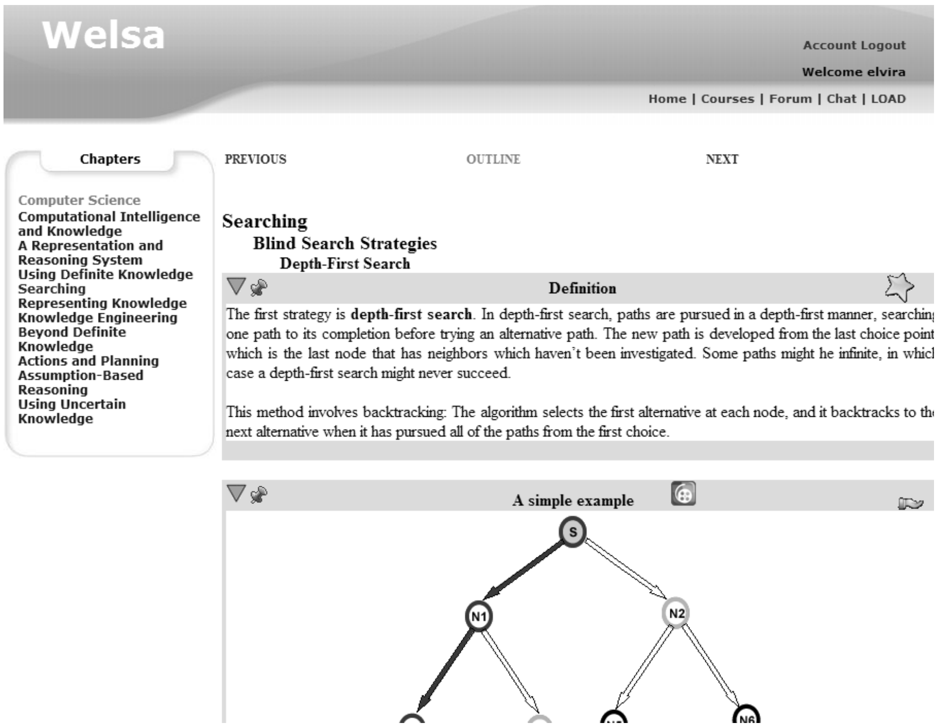
6. An AI course module implemented in WELSA. Based on the above approach, we have developed an educational hypermedia system called WELSA, which offers the following functionalities:

- an *authoring tool* for the teachers, allowing them to create courses conforming to the internal WELSA format, as described above;
- a *course player* for the students, enhanced with a learner tracking functionality (monitoring the student interaction with the system);
- an *analysis tool* allowing the researcher to interpret the behaviour of the student and identify the corresponding learning styles.

In order to validate our approach we have designed a course module in the area of Artificial Intelligence and implemented it in WELSA. The course module deals with search strategies and solving problems by searching and it is based on the fourth chapter of Poole, Mackworth and Goebel's AI textbook [24]. The course consists of 4 sections and 9 subsections, including a total of 46 learning objects (LOs). From the point of view of the media type, the course includes both *"text"* LOs (35), as well as *"image"*, *"video"* and *"animation"* LOs (11). From the point of view of the instructional role, the course consists of 12 *"Fundamental"* LOs (5 *"Definition"* and 7 *"Algorithm"*) and 34 *"Auxiliary"* LOs (4 *"Additional Info"*, 1 *"Demonstration"*, 14 *"Example"*, 5 *"Exercise"*, 3 *"Exploration"*, 5 *"Introduction"*, 1 *"Objectives"* and 1 *"Remark"*). The course also includes access to two communication tools, one synchronous (chat) and one asynchronous (forum) and offers two navigation choices—either by means of the *Next* and *Previous* buttons, or by means of the *Outline*.

Initially, only the first LO on each page is expanded, the rest being shown in a strechtext format, including only the resource title and some visual cues such as icons for the instructional role and the media type. However, the student has the possibility to expand any LOs on the page and "lock" them in the expanded format. She/he can thus choose between having several LOs available at the same time or concentrating on only a single LO at a time.

Figure 6.1 shows a snapshot from the "Blind Search Strategies" section, more specifically the Depth-First Search subsection. The fragment includes one LO with *LoType2="Definition"* and *dc:type="text"* and one with *LOType3="Example"* and *dc:type="animation"*, both in an expanded state.



The first strategy is **depth-first search**. In depth-first search, paths are pursued in a depth-first manner, searchin
one path to its completion before trying an alternative path. The new path is developed from the last choice point
which is the last node that has neighbors which haven't been investigated. Some paths might he infinite, in whic
case a depth-first search might never succeed.

This method involves backtracking: The algorithm selects the first alternative at each node, and it backtracks to th
next alternative when it has pursued all of the paths from the first choice.

FIG. 6.1. *A snapshot of WELSA course player*

Fragments of the corresponding XML files are included in the Appendix (`course.xml`, `chapter.xml`, `depth_first_definition.xml`).

7. Related Works. Currently there are several works that address aspects related to ontologies and metadata for personalized e-learning, such as: [1, 6, 13, 16, 17, 28]. A few of them, that we will briefly discuss here, also take into consideration learning styles.

In case of [16] the ontology is tied to a particular learning style model, namely Felder-Silverman (FSLSM) [14]. There is a special class, *LearningStyle*, which represents the FSLSM dimension associated to a particular learning object (active-reflective, visual-verbal, sensing-intuitive, sequential-global). Thus all learning objects have to be indexed according to FSLSM in order to allow for delivering of adapted content.

Paper [6] proposes a learning style taxonomy, based on Curry's onion model [11]. In the LAG adaptation model, each learning style can be associated with a specific instructional strategy, which can be broken down into adaptation language constructs, which in their turn can be represented by elementary adaptation techniques. It is the role of the author to specify not only the annotated learning content (the static description) but also the adaptation logic (the dynamic description).

Finally, paper [28] introduces the concept of Open Learning Objects, which represent distributed multimedia objects in SVG format. They incorporate inner metadata in XML format which is structured on several levels (content, adaptation, animation…). Each Open Learning Object is tied to a particular learning style dimension; however any learning style model can be employed, by configuring the adaptation markup.

8. Conclusions. In this paper we sketched an intelligent way of organizing the learning resources in an LSAES. Based on Dublin Core metadata [12] and Ullrich's instructional ontology [29], we introduced a set of educational metadata that are independent of any learning style. We then showed how these metadata can be employed for modelling the learner and applying various adaptation techniques. The approach was illustrated with a course module in the area of Artificial Intelligence, which was created and deployed using WELSA, our dedicated adaptive educational system. As future work, we intend to validate our approach through experimental research, evaluating WELSA in real-world settings.

Appendix. Examples of course, chapter and metadata files.

`course.xml.`

```xml
<?xml version="1.0" encoding="UTF-8"?>
<course xmlns:p="http://purl.org/dc/elements/1.1/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="course.xsd"
        xsi:schemaLocation="http://purl.org/dc/elements/1.1/
        http://dublincore.org/schemas/xmls/qdc/2006/01/06/dc.xsd">
  <About>
    <title>Artificial Intelligence</title>
    <identifier>CS104</identifier>
    <creator>Elvira Popescu</creator>
    <date>12-01-2008</date>
    <subject>Artificial Intelligence</subject>
    <description>This is an introductory course on AI</description>
    <language>en</language>
    <source>Computational Intelligence - D. Poole, A. Mackworth, R. Goebel</source>
    . . .
  </About>
  <Content>
    <Chapter number="1">Computational Intelligence and Knowledge</Chapter>
    <Chapter number="2">A Representation and Reasoning System</Chapter>
    <Chapter number="3">Using Definite Knowledge</Chapter>
    <Chapter number="4">Searching</Chapter>
    <Chapter number="5">Representing Knowledge</Chapter>
    <Chapter number="6">Knowledge Engineering</Chapter>
    <Chapter number="7">Beyond Definite Knowledge</Chapter>
    <Chapter number="8">Actions and Planning</Chapter>
    <Chapter number="9">Assumption-Based Reasoning</Chapter>
    <Chapter number="10">Using Uncertain Knowledge</Chapter>
  </Content>
</course>
```

`chapter.xml.`

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Chapter xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="chapter.xsd"
    xsi:schemaLocation="http://purl.org/dc/elements/1.1/
    http://dublincore.org/schemas/xmls/qdc/2006/01/06/dc.xsd">
    <About>
        <title>Searching</title>
        <creator>Elvira Popescu</creator>
    . . .
</About>
<Content>
    <Div1>
        <Title>Why Search?</Title>
        <Div2>
            <Title></Title>
            <Div3>
                <Title>Why Search?</Title>
                <Div4>
                    <Title></Title>
                    <LO>search_introduction.xml </LO>
                    <LO>search_def.xml </LO>
                    <LO>search_objectives.xml </LO>
                </Div4>
            </Div3>
        </Div2>
    </Div1>
    <Div1>
        <Title>Graph Searching</Title>
        <Div2>
            <Title></Title>
            <Div3>
                <Title>Graph Searching</Title>
                <Div4>
                    <Title></Title>
                    <LO>graph_search_introduction.xml</LO>
                    <LO>graph_search_example.xml</LO>
                </Div4>
            </Div3>
            <Div3>
                <Title>Formalizing Graph Searching</Title>
                <Div4>
                    <Title></Title>
                    <LO>fgraph_search_definition.xml</LO>
                    <LO>fgraph_search_example.xml</LO>
                    <LO>test_graph_searching.xml</LO>
                </Div4>
            </Div3>
        </Div2>
    </Div1>
    <Div1>
        <Title>A Generic Searching Algorithm</Title>
        <Div2>
            <Title></Title>
            <Div3>
                <Title>A Generic Searching Algorithm</Title>
                <Div4>
                    <Title></Title>
                    <LO>generic_search_introduction.xml</LO>
                    <LO>generic_search_example1.xml</LO>
                    <LO>generic_search_procedure1.xml</LO>
                    <LO>generic_search_example2.xml</LO>
                    <LO>generic_search_demonstration.xml</LO>
                    <LO>generic_search_addinf.xml</LO>
                    <LO>generic_search_costs.xml</LO>
                </Div4>
            </Div3>
            <Div3>
                <Title>Finding Paths</Title>
                <Div4>
                    <LO>generic_search_paths_introd.xml</LO>
                    <LO>generic_search_paths_procedure.xml</LO>
                    <LO>generic_search_paths_example.xml</LO>
                    <LO>test_generic_search.xml</LO>
                </Div4>
            </Div3>
        </Div2>
    </Div1>
    <Div1>
        <Title>Blind Search Strategies</Title>
        <Div2>
            <Title></Title>
            <Div3>
                <Title>Blind Search Strategies</Title>
                <Div4>
                    <LO> blind_strategies_introduction.xml</LO>
                    <LO> tutorial_text.xml</LO>
                    <LO> tutorial_video.xml</LO>
                </Div4>
            </Div3>
```

```
                <Div3>
                    <Title>Depth-First Search</Title>
                    <Div4>
                        <LO> depth_first_definition.xml</LO>
                        <LO> depth_first_simple_example.xml</LO>
                        <LO> depth_first_procedure.xml</LO>
                        <LO> depth_first_example_txt.xml</LO>
                        <LO> depth_first_example_anim.xml</LO>
                        <LO> depth_first_example_sim.xml</LO>
                        <LO> depth_first_addinf.xml</LO>
                        <LO> test_DFS.xml</LO>
                    </Div4>
                </Div3>
                <Div3>
                    <Title>Breadth-First Search</Title>
                    <Div4>
                        <LO> breadth_first_definition.xml</LO>
                        <LO> breadth_first_simple_example.xml</LO>
                        <LO> breadth_first_procedure.xml</LO>
                        <LO> breadth_first_example_txt.xml</LO>
                        <LO> breadth_first_example_anim.xml</LO>
                        <LO> breadth_first_example_sim.xml</LO>
                        <LO> breadth_first_addinf.xml</LO>
                        <LO> test_BFS.xml</LO>
                    </Div4>
                </Div3>
                <Div3>
                    <Title>Lowest-Cost-First Search</Title>
                    <Div4>
                        <LO> lcost_first_definition.xml</LO>
                        <LO> lcost_first_simple_example.xml</LO>
                        <LO> lcost_first_procedure.xml</LO>
                        <LO> lcost_first_example_txt.xml</LO>
                        <LO> lcost_first_example_anim.xml</LO>
                        <LO> lcost_first_example_sim.xml</LO>
                        <LO> lcost_first_addinf.xml</LO>
                        <LO> test_LCFS.xml</LO>
                    </Div4>
                </Div3>
            </Div2>
        </Div1>
    </Content>
</Chapter>
```

`depth_first_definition.xml`.

```
<?xml version="1.0" encoding="UTF-8"?>
<LO xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="metadata.xsd"
    xsi:schemaLocation="http://purl.org/dc/elements/1.1/
    http://dublincore.org/schemas/xmls/qdc/2006/01/06/dc.xsd">
    <title>Definition</title>
    <identifier>depth_first_definition.html</identifier>
    <type>text</type>
    <format>text/html</format>
    . . .
    <LoType1>Fundamental</LoType1>
    <LoType2>Definition</LoType2>
    <hasAbstractness>neutral</hasAbstractness>
    <hasFormalness>informal</hasFormalness>
</LO>
```

REFERENCES

[1] Al-Khalifa H. S., Davis H. C.: The Evolution of Metadata from Standards to Semantics in E-Learning Applications. *Proceedings of Hypertext'06,* Odense, Denmark (2006).

[2] ADL SCORM. `http://www.adlnet.gov/scorm/index.aspx`

[3] ALFANET—D8.2 Public final report. `http://rtd.softwareag.es/alfanet/` (2005).

[4] Anido L., Fernandez M., Caeiro M., Santos J., Rodriguez J., Llamas M.: Educational metadata and brokerage for learning resources. *Computers and Education*, Elsevier, 38 (2002) 351–374.

[5] Bajraktarevic N., Hall W., Fullick P.: Incorporating learning styles in hypermedia environment: Empirical evaluation. *Procs. Workshop on Adaptive Hypermedia and Adaptive Web-Based Systems* (2003) 41–52.

[6] Brown E., Cristea A., Stewart C., Brailsford T.: Patterns in Authoring of Adaptive Educational Hypermedia: A Taxonomy of Learning Styles. *Educational Technology and Society*, 8, (2005) 77–90.

[7] Brusilovsky P.: Adaptive navigation support: from adaptive hypermedia to the adaptive Web and beyond. *PsychNology Journal,* Vol. 2, No. 1 (2004) 7–23.

[8] Carver C. A., Howard R. A., Lane W. D.: Enhancing student learning through hypermedia courseware and incorporation of student learning styles. *IEEE Transactions on Education*, 42 (1999) 33–38.
[9] Cha H. J., Kim Y. S., Park S. H., Yoon T. B., Jung Y. M., Lee J. H.: Learning styles diagnosis based on user interface behaviors for the customization of learning interfaces in an intelligent tutoring system. *Proc. 8th Intl. Conf. on Intelligent Tutoring Systems*, LNCS, Vol. 4053, Springer (2006).
[10] Coffield F., Moseley D., Hall E., Ecclestone K.: *Learning styles and pedagogy in post-16 learning. A systematic and critical review.* Learning and Skills Research Centre, UK (2004).
[11] Curry L.: *Integrating Concepts of Cognitive or Learning Style: A Review with Attention to Psychometric Standards.* Ottawa, ON: Canadian College of Health Service Executives (1987).
[12] Dublin Core Metadata Initiative `http://dublincore.org/`
[13] Dolog P., Henze N., Nejdl W., Sintek M.: Personalization in distributed eLearning environments, *Proc. Intl. Conf. WWW2004* (2004).
[14] Felder R. M., Silverman L. K.: Learning and teaching styles in engineering education. *Engineering Education*, Vol. 78, No. 7 (1988).
[15] Gardner H.: *Multiple Intelligences: The Theory in Practice.* Basic Books, New York (1993).
[16] Gascuena J., Fernández-Caballero A., Gonzalez P.: Domain Ontology for Personalized E-Learning in Educational Systems. *Proc. 6th IEEE Intl. Conf. on Advanced Learning Technologies—ICALT 2006*, IEEE Computer Society Press (2006).
[17] Geser G. (Ed.): *OLCOS Roadmap 2012—Open Educational Practices and Resources*, Austria (2007).
[18] Honey P., Mumford A.: *The learning styles helper's guide.* Maidenhead: Peter Honey Publications Ltd. (2000).
[19] IEEE LOM `http://ltsc.ieee.org/wg1`
[20] IMS MD: `http://www.imsglobal.org/metadata/index.html`
[21] Keefe J. W.: Learning style: an overview. *NASSP's Student Learning Styles: Diagnosing and Prescribing Programs* (1979) 1–17.
[22] Kelly D., Tangney B.: Adapting to intelligence profile in an adaptive educational system. *Interacting with Computers,* Elsevier, 18 (2006) 385–409.
[23] Papanikolaou K. A., Grigoriadou M., Kornilakis H., Magoulas G. D.: Personalizing the interaction in a Web-based educational hypermedia system: the case of INSPIRE. *User-Modeling and User-Adapted Interaction*, 13 (2003) 213–267.
[24] Poole D., Mackworth A., Goebel R.: *Computational Intelligence: A Logical Approach.* Oxford University Press (1998).
[25] Popescu E., Bădică C., Trigano P.: Rules for Learner Modeling and Adaptation Provisioning in an Educational Hypermedia System. *Proc. SYNASC 2007 (workshop RuleApps)*, IEEE Computer Society Press (2007), 492–499.
[26] Popescu E., Trigano P., Bădică C.: Towards a Unified Learning Style Model in Adaptive Educational Systems. *Proc. 7th IEEE Intl. Conf. on Advanced Learning Technologies—ICALT 2007*, IEEE Computer Society Press (2007), 804–808.
[27] Popescu E., Trigano P., Bădică C.: Adaptive Educational Hypermedia Systems: A Focus on Learning Styles. *Proc. 4th IEEE Intl. Conf. on Computer as a Tool—EUROCON 2007*, IEEE Press (2007), 2473–2478.
[28] Shi H., Rodriguez O., Chen S., Shang Y.: Open Learning Objects as an Intelligent Way of Organizing Educational Material. *International Journal on E-Learning*, 3(2) (2004) 51–63.
[29] Ullrich C.: The Learning-Resource-Type is Dead, Long Live the Learning-Resource-Type!. *Learning Objects and Learning Designs*, 1 (2005) 7–15.

Edited by: Maria Ganzha, Marcin Paprzycki
Received: March 23, 2008
Accepted: March 28, 2008

Scalable Computing: Practice and Experience
Volume 9, Number 1, pp. 21–28. http://www.scpe.org

ISSN 1895-1767
© 2008 SCPE

A LINK-CLUSTER ROUTE DISCOVERY PROTOCOL FOR AD HOC NETWORKS

DOINA BEIN*, AJOY K. DATTA†, AND SHASHIREKHA YELLENKI†

Abstract. In MANETS, node mobility induces structural changes for routing. We propose a route discovery algorithm for MANET based on the link-cluster architecture. The algorithm selects the clusterheads and gateway nodes, and then builds routing tables for nodes both inside and outside the cluster. The algorithm attempts to minimize the number of clusterheads and gateway nodes to avoid storing redundant data. For intra-cluster routing, the shortest paths are maintained. For inter-cluster routing, we implement routing on-demand (the shortest paths are maintained only for the nodes that need to send packets). The proposed algorithm adapts to arbitrary movement of nodes, and joining and/or leaving of existent nodes.

Key words. ad hoc networks, clustering, location management, agent mobility

1. Introduction.

A mobile ad-hoc network (MANET) is a self-configuring network of mobile hosts connected by wireless links, with an arbitrary topology. The mobility management of such networks is important since a minimal configuration and quick deployment make ad hoc networks suitable for emergency situations like natural or human-induced disasters, military conflicts, emergency medical situations, etc. Beginning as a military application, MANETs had become largely used for personal use, e.g. personal area network (PAN) (for short-range communication of user devices), wireless local area network (WLAN) and in-house digital network (IHDN) (for video and audio data exchange).

The first self-configuring (self-organizing) protocols for a MANET (protocol LCA [1, 2], protocol DEA [15], protocol Layer Net [3]) periodically discard the network topology information and rebuild the network from scratch. Later protocols consider a gradual approach to self-configure a MANET (for example, the protocol SWAN by Scott and Bambos [17]).

The first self-configuring wireless network, proposed by Baker and Ephremides [1, 2], is a two-tier hierarchical model. The nodes, classified as ordinary, clusterheads, and gateways, have the restriction that a node belongs to a single cluster (clusterhead) and it is one hop away from it. Since selecting the minimum number of such clusterheads is NP hard, they proposed a link cluster algorithm (LCA) for categorizing the nodes and a link activation algorithm (LAA) to schedule (activate) the links between nodes. LCA algorithm is a dominating set partitioning of the network based on node ID and works as follows. The node with the highest identity number among a group of nodes without a clusterhead within one hop declares itself as a clusterhead. The other nodes become either gateways (if there are connected to two or more clusterheads) or ordinary nodes.

Variations of the LCA algorithm are to either consider the lowest ID or the highest connected node instead of the highest ID node.

The distributed evolutionary algorithm (DEA) proposed by Post *et al.* [15] is based on a clique partitioning of the network (also an NP hard problem) and is uniform (it is the same for each node in the network). It works as follows. A starter node activates all its neighbors that are part of some clique as itself (so called *clique neighbors*) to begin communication based on a schedule decided by itself. Then these nodes become starter nodes for the rest of the network.

In protocol SWAN proposed by Scott and Bambos [17], new connections are sought during random access periods. After a timeout, the connections that do not respond to a control call are declared unusable.

In spite of the various applications served by the ad-hoc networks, they still have to overcome aspects as the limited transmission range, interference due to its broadcast nature, route changes and packet losses due to the node mobility, battery constraints, and potentially frequent network partitions. A major challenge faced in MANETs is locating the devices for communication, especially with high node mobility and sparse node density. Present solutions provided by the ad hoc routing protocols range from flooding [11] the entire network with route requests, to deploying a separate location management scheme [14] to maintain a device location database. Kawadia et al. [10] had given a general framework to support the implementation of ad-hoc routing protocols in Unix-like operating systems.

*Department of Computer Science, Erik Jonsson School of Engineering, University of Texas at Dallas, 2601 North Floyd Road, Richardson, TX 75083-0688 (siona@utdallas.edu).

†School of Computer Science, University of Nevada, Las Vegas, 4505 Maryland Parkway, Las Vegas, NV 89154-4019 ({datta, rekha}@cs.unlv.edu)

Contributions. We present a protocol for routing in ad hoc networks that adapts fast to frequent node movement, yet requires little or no overhead during periods in which hosts move less frequently. Moreover, the protocol routes packets through a dynamically established and nearly optimal path between two wireless nodes. It also achieves higher reliability—if a node in a cluster fails, the data is still accessible via other cluster nodes.

In a network with link-cluster architecture we propose a protocol that discovers an optimal route for the nodes to communicate. We use the concept of *proactive* protocols to route the packets within the cluster and the concept of *reactive* protocols to route the packets between the clusters. (A combination of proactive and reactive protocols used for routing the packets is called a *hybrid* protocol [16]). When a node leaves a cluster we update the routing tables (location management, [9]).

Outline of the paper. In Section 2 we present the architectural model and the variables used by the algorithm. The cluster-based route discovery algorithm is presented in Section 3, together with a proof of correctness in Section 4. We finish with concluding remarks in Section 5.

2. Preliminaries. Clustering is a scheme designed for large dynamic networks to build a control structure that increases network availability, reduces the delay in responding to changes in network state, and improves data security. Clustering is crucial for scalability as the performance can be improved by adding more nodes to the cluster.

Link-cluster architecture [1, 2, 8] is a network control structure in which nodes are partitioned into clusters that are interconnected. The union of the members of all the clusters covers all the nodes in the network. Nodes are classified into clusterheads, gateways, and ordinary nodes. A *clusterhead* schedules the transmissions and allocates resources within clusters. *Gateways* connect adjacent clusters. An *ordinary node* belongs to a single cluster (has a unique clusterhead). We will consider only *disjoint* clusters. Specifically, a gateway node is a member of exactly one cluster and forms links to members of other clusters.

A non-clusterhead node is within two hops from its clusterhead. Since there are no adjacent clusterheads, the clusterheads form an independent set of nodes.

For example, in Figure 2.1, an ad hoc network is divided into five clusters.

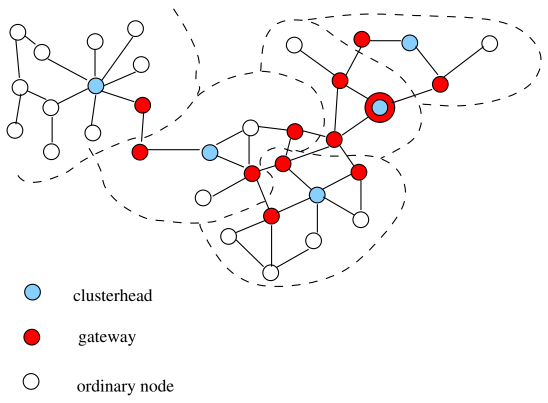


Fig. 2.1. *Ad hoc network divided into 5 clusters*

There are couple of advantages to such an architecture. Scalability is improved since a reduced number of mobile nodes participate in some routing algorithm, hence a low routing-related control overhead. Also, the chance of interference via coordination of data transmissions is lower.

Cluster maintenance schemes are designed to minimize the number of changes in the set of existing clusters. They do not re-cluster after every movement, but instead make small adjustment to the cluster membership as necessary, as in only when the most highly connected node in a cluster moves. All the gateway nodes and the clusterhead node which are present in the cluster region C_u of the clusterhead node u act as location servers for all the nodes in the cluster region C_u. When a node moves across two clusterhead regions, the node updates its home region C_u of the movement by a location update or by sending a leave message. A source node x from outside the cluster that needs to communicate with another node y in the cluster region C can use the clusterhead and gateway tables to identify the location of y and send a location query packet towards region C to obtain the current location of y. The first location server to receive the query for u responds with the current location of y to which data packets are routed.

3. Cluster-Based Route Discovery Algorithm.

Our algorithm is based on the fact that every mobile node has a unique identifier [7] that differentiates every single node in the network from others. The node with the highest identifier within a geographical region becomes the clusterhead [5].

The change in clusterhead status occurs only if two clusterheads move within the range of each other—in that case one of them relinquishes its role as clusterhead—or if an ordinary node moves out of range of all other nodes—in that case it becomes the clusterhead of its own cluster.

The algorithm uses a variable N_i^1 representing the one-hop neighborhood set of node i and a variable N_i^2 representing the two-hop neighborhood set of node i. These two sets are maintained by a local topology maintenance protocol that adjusts its value in case of topological changes in the network due to failures of nodes or links.

Node i has a unique ID, $ID.i$. Variable $n.i$ is used to identify the neighbor of the shortest path to the clusterhead (for a non-clusterhead node).

Every node in a network has a sequence table that keeps track of the messages already received by the node and makes the routing messages loop-free [4, 13]. Only gateways and clusterheads maintain the routing tables used for routing [6]. A clusterhead has another table that is used to route messages outside the cluster. This table has entries of all the destination and boundary gateway pairs. The gateway tables contain all the entries of the destination-clusterhead pairs of all the clusters they connect to. The routing table is updated whenever a new clusterhead is elected or some changes occur related to paths in the routing table. The ordinary nodes have only a variable indicating the neighbor on the shortest path towards their clusterhead.

The proposed protocol consists of three main modules: Clusterhead Election, Gateway Election, and Route Discovery.

3.1. Clusterhead Selection Module.

The clusterhead selection protocol must satisfy two conditions.
Condition 1: Each non-clusterhead is within two hops from its clusterhead.
Condition 2: There are no adjacent clusterheads [12].

A node can act as a clusterhead as well as a gateway at the same time.

A clusterhead will periodically do the following:

- It checks the consistency of each variable.
- It broadcasts CL_ANN messages to all its neighbors within its two hop distance.
- It checks whether any other clusterhead is within its transmission range and if it finds one whose ID is bigger than itself, then it gives up its clusterhead status by broadcasting CL_REJ messages.

Algorithm 3.1 Clusterhead Selection Module

```
Actions of some node $i$

E.01  Timeout  $\longrightarrow$
          if $i$ is a clusterhead then sends $CL\_ANN$ to immediate neighbors
          else if $i$ finds itself with faulty values or is "orphan" (has no CH),
              then elects itself as a clusterhead
              else $i$ sends $CL\_REQ$ message to $n.i$

E.02  Receive $CL\_ANN$ from node $nb$  $\longrightarrow$
          if $i$ is an ordinary node and either the sender was its own CH or
              $i$ has no current clusterhead, then updates its variable with respect
              to the sender as a clusterhead and forwards the message
          else if $i$ is a clusterhead and the sender is a clusterhead with
              a lower ID and within 2 hops, then $i$ accepts the sender as
              a clusterhead and sends $CL\_REJ$ to all neighbors

E.03  Receive $CL\_REJ$ from node $nb$  $\longrightarrow$
          if $i$ is a clusterhead, then drops the message
          else if the sender is $i$'s CH, then mark itself as "orphan" and forward it
```

Algorithm 3.2 Clusterhead Selection Module (continued)

```
E.04  Receive $CL\_REQ$ from node $nb$  $\longrightarrow$
          if $i$ is a clusterhead then
              if the sender belongs to its cluster, then send $CL\_ANN$ to sender
              else send $CL\_CHG$ to the sender
          else if the message is addressed to $i$ then reply with $CL\_CHG$
              else if the addressee is within two hops, then forward it to addressee
                  else drop the message

E.05  Receive $CL\_CHG$ from node $nb$  $\longrightarrow$
          if the message is regarding is $i$'s clusterhead, then $i$ updates
              its variables accordingly and forwards the message to neighbors

E.06  Receive $CL\_ACCEPT$ from node $nb$  $\longrightarrow$
          if $i$ is a clusterhead and the addressee, then updates its routing table
              and sends the updated message to the bordering gateway nodes
          else if the message is not addressed to the node, then it forwards
              the message to its neighbors if the hop count $< 2$,
              but drops the message if the hop count $\geq 2$

E.07  Receive $leave$ from node $nb$  $\longrightarrow$
          if $i$ is a clusterhead and the addressee, then updates its routing table
              and sends the updated message to the bordering gateway nodes
          else if the message is not addressed to the node, then it forwards
              the message to its neighbors if the hop count $< 2$,
              but drops the message if the hop count $\geq 2$

E.08  Receive $ctable\_copy$ from node $nb$  $\longrightarrow$
          if $i$ is a clusterhead and the message is addressed to it, then the row
              contained in the message is copied into the routing table if
              the destination node is within 2 hop distance

E.09  Receive $CL\_CHG$ from node $nb$  $\longrightarrow$
          if $i$ is gateway and the sender is clusterhead of one of its neighbors, then
              updates its $GC\_TABLE$
```

An ordinary node periodically checks whether its clusterhead is still alive or not, by sending a CL_REQ message through $n.i$. In case it finds out that it has no clusterhead within two hop distance, then it sets its variables accordingly and waits for a CL_ANN message from a clusterhead node within two hops distance. The ordinary node becomes a clusterhead if there is no clusterhead within two hops distance.

A CL_REQ message travels at most two hops from the sender. Once the CL_REQ message reaches the right destination but finds that the clusterhead moved from that location, the node in that particular location or the node which was supposed to be the one hop neighbor on the shortest path from the sender to the supposed-to-be clusterhead's location sends a CL_CHG message indicating that the previous clusterhead no longer exists in that location.

3.2. Gateway Selection Module. In the gateway selection protocol, a gateway node periodically does the following. It checks whether there exists another gateway in two hop distance that connects the same clusters. If it finds one, it compares its own ID with it. If it has a smaller ID, then it relinquishes its role as a gateway by updating its $g.i$ variable and sending a GW_REJ message.

3.3. Route Discovery Module. For route discovery, we have intra-cluster (routing within the cluster) and inter-cluster routing (routing between the clusters).

For intra-cluster routing, each clusterhead keeps in its routing table data about the nodes that belong to its own cluster, collected in the clusterhead election module using CL_REQ messages. These messages

Algorithm 3.3 Gateway Selection Module

```
Actions of some node $i$

G.01  Timeout     $\longrightarrow$ if $i$ is a gateway and there is another gateway within
          2 hops with a lower ID that connects at least the clusters, then sends
          $GL\_REJ$ to all neighbors

G.02  Receive $GL\_ANN$ from node $nb$     $\longrightarrow$
          if $i$ is a clusterhead and the message is addressed to it, then updates
              its inter-cluster table
          else if $i$ is a gateway and there is another gateway within 2 hops with
              a lower ID that connects at least the clusters, then $i$ sends
              $GL\_REJ$ to all neighbors
              else it forwards the message to its neighbors if the hop count $< 2$,
                  but drops the message if the hop count $\geq 2$

G.03  Receive $GW\_REJ$ from node $nb$     $\longrightarrow$
          if $i$ is a clusterhead and the message regards one of its bordering
              gateway node, it removes all such rows containing the sender's ID
              in the GW field of its tables
          else it forwards the message to its neighbors if the hop count $< 2$,
                  but drops the message if the hop count $\geq 2$
```

are periodically sent by a non-clusterhead node to check the status of its own clusterhead and the path towards it.

For inter-cluster routing, the clusterheads as well as the gateway nodes keep track of the gateway-destination and clusterhead-destination pairs, respectively, to reach the temporary destination, which is a milestone in reaching the actual destination. This data is collected only when there is a need to communicate with the node and stored in the inter-cluster tables. The tables are purged by the routes that are unused for a long time, and their entries are kept up-to-date.

The following steps are repeated until the route is found.

1. Sender checks with its clusterhead if its routing table has an entry for the destination node that it wants to communicate with. If the cluster-head has an entry, the sender gets the path from the clusterhead and uses it to communicate.
2. If the clusterhead's routing table does not have an entry, it checks with the clusterhead's gateway table. If it finds an entry, then it uses that route to communicate.
3. If the clusterhead's gateway table does not have an entry, then it checks with the gateway's cluster tables of all the bordering gateways for the route. If it finds the route, it uses that to communicate.

4. Proof of Correctness. Lemma 4.1. *The maximum number of hops between a clusterhead and a member of its own cluster is two.*

Proof. In clusterhead election module, Actions E.02 and E.06 ensure that any clusterhead announcement (CL_ANN) message or the clusterhead accept (CL_ACCEPT) message can travel at most a distance of two hops. For a node to be a member of a cluster it has to receive the clusterhead announcement message from a clusterhead and send the clusterhead accept message back to the clusterhead, which is possible only if the node is at a two-hop distance from its clusterhead. □

Lemma 4.2. *No two clusterheads can be neighbors of each other.*

Proof. We prove this lemma by contradiction. Suppose there are two clusterheads that are neighbors.

Action E.02 ensures that the clusterhead announcement message (CL_ANN) of one clusterhead reaches the other that is at one or two-hop distance from it (Lemma 4.1). When a clusterhead receives a clusterhead announcement message, it compares its own ID with the sender's ID. If its ID is less than the sender's ID, it relinquishes its role as a clusterhead and sends a clusterhead reject message (CL_REJ) message to all its two-hop neighbors.

Algorithm 3.4 Route Discovery Module

Actions of some node i

```
A.01  Receive *Routedisc* from node $nb$   $\longrightarrow$
          if the same message was received before, then drop it
          if $i$ is a clusterhead
              if the message was addressed to $i$ then sends back an *ack* message
              else if the destination node belongs to its cluster, it sends
                  the *shortestpath* message to the sender
                  else it updates its inter-cluster table and sends the
                      updated message to the bordering gateway nodes
          else if $i$ is a gateway
              if the message was addressed to $i$ then sends back an *ack* message
              else if the destination node belongs to its inter-cluster table, it
                  forwards it to all the clusterheads in its inter-cluster table
                  else it updates its inter-cluster table and sends the
                      updated message to the bordering gateway nodes
          else if $i$ is an ordinary node
              if the message was addressed to $i$ then it sends back an *ack* message
              else forwards the message to its neighbors

A.02  Receive *me_dest* from node $nb$   $\longrightarrow$
          if $i$ is the clusterhead of the destination and the sender does not belong to
              its inter-cluster routing table, it updates the table and sends the updated
              message to all its bordering clusterheads
          else if $i$ is a gateway
              if the clusterhead of the destination is at one hop distance,
                  it forwards the message
              if the sender does not belong to the inter-cluster routing table,
                  it updates the table and sends the updated message to
                  all its bordering clusterheads
          else if $i$ is an ordinary node and the clusterhead of the destination is
              at one hop distance, it forwards the message

A.03  Receive *me_dest* from node $nb$   $\longrightarrow$
          if $i$ is a clusterhead or a gateway
              if the sender does not belong to its inter-cluster routing table,
                  it updates its table and sends the updated message to all its
                  bordering gateway nodes
              if it is not the destination, then it forwards the message to all
                  the nodes in the specified in the field *route* of the message
          else if $i$ is an ordinary node and it is not the destination, then it forwards
              the message to all nodes in the specified in the field *route* of the message

A.04  Receive *ack* from node $nb$   $\longrightarrow$
          if $i$ is a clusterhead, it updates its table and sends the updated message
              to the bordering gateways
          else if $i$ is a gateway, it updates its table and sends the updated message
              to the bordering clusterheads
          else if $i$ is an ordinary node, if the clusterhead of the destination is at
              one hop distance, it forwards the message to that particular neighbor
```

Action E.03 ensures that a clusterhead reject message reaches all the two-hop neighbors. So, the clusterhead with lower ID no longer remains a clusterhead. This contradicts our assumption that there can be two clusterheads that can be one-hop neighbors. □

LEMMA 4.3. *The minimum number of hops between two clusterheads is three.*

Algorithm 3.5 Route Discovery Module (continued)

```
A.05  Receive Ctable_update from node nb      ⟶
          if i is a gateway
              if the message is from a neighboring clusterhead, it updates its
                  inter-cluster routing table, else forwards it to its neighbors
          else if i is an ordinary node, not the addressee, but the addressee is
              a neighbor then it forwards the message to it

A.06  Receive Gtable_update from node nb      ⟶
          if i is a clusterhead
              if the message is from a gateway node that is present in its inter-cluster
                  routing table, it updates its inter-cluster routing table
              else forwards it to its neighbors
          else if i is an ordinary node, not the addressee, but the addressee is
              a neighbor then it forwards the message to it
```

Proof. From Lemma 4.2, no two clusterheads can be neighbors of each other. Assume that the distance between two clusterheads is two hops. But because the node between them becomes a gateway and acts as a common node for both clusters, that cancels one of the two clusterheads with lower ID. □

LEMMA 4.4. *The maximum number of hops between the clusterheads of two neighboring clusters is five.*

Proof. Let us assume that the distance between two given clusterheads is six. According to *Clusterhead Selection* Module, Action E.02 ensures that any clusterhead announcement message travels at most a distance of two hops. Then, there is at least one node situated in between the two clusterheads that does not receive any clusterhead announcement message. This node waits for a timeout period (Action E.01) and then, at timeout, it sets itself a clusterhead forming its own cluster. Then the distance between the two original clusterheads reduces to three. □

LEMMA 4.5. *If there exists only one link connecting two neighboring clusters then the eligible gateway node(s) of the link will be selected as gateway nodes.*

Proof. We prove this lemma by contradiction. Suppose the nodes connecting the clusters are not gateway nodes. By the definition of a gateway, both nodes are eligible gateway nodes because both of them have at least one neighbor that does not belong to its own cluster. In *Gateway Selection* Module, we eliminate the eligible gateway nodes becoming the gateway nodes only if they belong to the same cluster. So, both the nodes become the gateway nodes that contradict the assumption that they are not gateway nodes. □

LEMMA 4.6. *If both the sender and destination are in the same cluster, a route discovery message is always acknowledged.*

Proof. When a node generates a route discovery message (*Routedisc*), it first sends it to its own clusterhead. Route discovery within a cluster means that the sender and destination belong to the same cluster. If the message reaches the destination before reaching the clusterhead, the destination node directly sends the acknowledgment (*ack*) message to the sender following the reverse path followed by the route discovery message. If the message reaches the clusterhead, all the clusterheads have entries for all the nodes in their intra-cluster table (routing table as named in *Route Discovery* Module) that belong to its own cluster. Once the clusterhead receives the message, it looks in its routing table, attaches the route from itself to the destination to the path followed by the route discovery message, and sends an acknowledgment message to the sender using a *shortestpath* message on the reverse path followed by the route discovery message.□

LEMMA 4.7. *If a node moves to another cluster, the route discovery algorithm will be able to find the node in finite time upon a request.*

Proof. When a node is part of a cluster, it periodically acknowledges a clusterhead that it is still part of the cluster.

When the node moves out of the cluster, the clusterhead waits for a timeout interval, then removes all the rows with this node as destination from its intra- and inter-cluster routing tables, and updates the same to its boundary gateway nodes so that they can remove the rows from their inter-cluster routing tables.

If the node joins another cluster, it acknowledges the new clusterhead's CL_ANN message with a CH_ACCEPT message, to acknowledge that it has joined the new cluster. The new clusterhead updates its entry in its intra-cluster routing table.

If the node itself becomes the clusterhead because it is not in two-hop distance from any clusterhead, then it broadcasts CL_ANN messages to all the other nodes. Eventually gateways nodes adjacent to that cluster will receive the message and the route is thus discovered. $\square$

5. Conclusion. We have presented a route discovery algorithm for MANET based on link-cluster architecture. The algorithm selects the clusterheads and gate-way nodes, and then builds routing tables for nodes both inside and outside the cluster. The proposed protocol guarantees that in finite number of steps, the network is divided into clusters. The algorithm attempts to minimize the number of clusterheads and gateway nodes to avoid storing redundant data. For intra-cluster routing, the shortest paths are maintained. For inter-cluster routing, we implement routing on-demand (the shortest paths are maintained only for the nodes that need to send packets). For both inter- and intra-cluster routing, the paths are loop free.

The proposed algorithm adapts to arbitrary movement of nodes, and joining and/or leaving of existent nodes.

As future work, we currently explore the possibility of a self-stabilizing cluster-based route discovery, in which, starting from an arbitrary configuration of the network, a correct configuration is reached in finite time without human intervention.

Shortest paths are guaranteed only for intra-cluster routing. Another direction for future research is to study the degree of sub-optimal paths for inter-cluster routing by varying various parameters.

REFERENCES

[1] D. J. Baker and A. Ephremides: A Distributed Algorithm for Organizing Mobile Radio Telecommunication Networks. *Proceedings of the Second International Conference on Distributed Computer Systems*, pages 476–483, April 1981.

[2] D. J. Baker and A. Ephremides: The Architectural Organization of a Mobile Radio Network via a Distributed Algorithm. *IEEE Transactions on Communications*, 29(11), pages 1694–1701, November 1981.

[3] A. Bhatnagar and T. G. Robertazzi: Layer Net: a New Self-organizing Network Protocol. *Military Communications Conference (Milcom)*, vol. 2, pages 845–849, 1990.

[4] G. G. Chen, J. W. Branch, B. K. Szymanski: Self-selective routing for wireless ad hoc networks. *IEEE International Conference on Wireless And Mobile Computing, (WiMob'2005)*, August 2005.

[5] C. C. Chiang: Routing in Clustered Multihop, Mobile Wireless Networks. *Proceedings of the ICOIN*, 1996.

[6] C. C. Chiang, H-K Wu, W. Liu, and M. Gerla: Routing in Clustered Multihop, Mobile Wireless Networks. *IEEE Singapore International Conference on Networks*, pages 197–211, 1997.

[7] S. R. Das, C. E. Perkins, and E. M. Royer: Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks. *Proceedings of INFOCOM*, March 2000.

[8] A. Ephremides, J. E. Wieselthier, and D. J. Baker: A Design Concept for Reliable Mobile Radio Networks with Frequency Hopping Signaling. *Proceedings of the IEEE*, 75(1), pages 56–73, January 1987.

[9] Z. Kai, W. Neng, and L. Ai-Fang: A new AODV based clustering routing protocol. *International Conference on Wireless Communications, Networking, and Mobile Computing*, September 2005.

[10] V. Kawadia, Y. Zhang, and B. Gupta: System Services for Implementing Ah-hoc Routing Protocols. *Proceedings of the International Conference on Parallel Processing Workshops (ICPPW'02)*, pages 135–142, 2002.

[11] Y. B. Ko and N. H. Vaidya: Location-aided routing in mobile ad hoc networks. *Technical report 98-012, Texas A&M University*, 1998.

[12] T. Johansson, L. Carr-Motyckova: Bandwidth-constrained Clustering in Ad Hoc Networks *The Third Annual Mediterranean Ad Hoc Networking Workshop*, pages 379–385, June 2004.

[13] C. E. Perkins and E. M. Royer: Ad-Hoc On-Demand Distance Vector Routing. *Second Annual IEEE Workshop on Mobile Computing Systems and Applications*, pages 99–100, February 1999.

[14] S. J. Philip, J. Ghosh, S. Khedekar, and C. Qiao: Scalability analysis of location management protocols for mobile ad hoc networks. *Wireless Communications and Networking Conference*, March 2004.

[15] M. J. Post, S. Kershenbaum, and P. E. Sarachik: A Distributed Evolutionary Algorithm for Reorganizing Network Communication. *Military Communications Conference (Milcom)*, 1985.

[16] E. M. Royer and C. K. Toh: A Review of Current Routing Protocols for Ad hoc Mobile Networks. *IEEE Personal Communications*, 6(2), pages 46–55, April 1999.

[17] K. Scott and N. Bambos: Formation and Maintenance of Self-Organizing Wireless Networks. *Thirty-First Asilomar Conference on Signals, Systems, and Computers*, vol. 1, pages 31–35, 1997.

Edited by: Maria Ganzha, Marcin Paprzycki
Received: Jan 30, 2008
Accepted: Feb 9, 2008

Scalable Computing: Practice and Experience
Volume 9, Number 1, pp. 29–37. http://www.scpe.org
ISSN 1895-1767
© 2008 SCPE

THE DILEMMA OF TRUST: A SOCIAL NETWORK BASED APPROACH

V. CARCHIOLO, A. LONGHEU, M. MALGERI, G. MANGIONI, V. NICOSIA[*]

Abstract. Trust is essential in human interaction and this naturally reflects within computer science context, in particular inside internet-based communities. In this work we present an approach based on social networks, which revealed several useful properties, as the *small world* effect, that can be usefully exploited in addressing the question of trust. We aim at reproducing the behaviour individuals adopt in their life when they establish and maintain trust relationships, sending queries to collect reputations in order to estimate how much trust in new acquaintances. We also consider issues as query forwarding and lifecycle of trusting relationships, aiming at building an effective and efficient model for trust management.

Key words. trusting and reputation, social networks, p2p networks

1. Introduction.

Trust is a crucial matter in several human interactions that must be addressed in order to cope with uncertainty when living and working with others. The question naturally reflects within the computer science context, specifically inside Internet-based communities, where the spread of networks and sharing of information over last decades (as in P2P paradigma) imposes to address the dilemma of trust, that can be viewed as an extension of trusting among people.

Several different definitions can be found for trust, e.g. in [13, 17, 1, 5]; a simple and effective definition is "trust provides information about with whom we should share information, from whom we should accept information, and what consideration such information should be given".

Studies on social networks over last years revealed that relationships are characterized by useful properties, as the "small world" effect, according to which a generic person is connected to another one through short paths of social relationships. This mechanism can be positively exploited when addressing the question of trust, in particular by emulating the behaviour individuals adopt in the real world to evaluate how much to trust others.

In our work, we consider how a person joins an existing network of trust relationships in real life, and how he builds these relationships over time, in particular reproducing the mechanism of collecting and mediating other's opinion about an unacquainted person to assign him a trust value. Our proposal falls within the context of *reputation-based* systems [7, 3], allowing an estimation of trust, in opposition to *policy-based* systems, where the hard evidence of owned credentials is used to grant trust[1]. We also consider trust relationships lifecycle, i.e. the evolution from "weak" (first time impressions) to "strong" (experience based judgement) links, as well as the removal of links between individuals not having contacts for long periods, and the effect of "feedback", that is the propagation of trust value from a person to neighbours, in order to inform them about a positive or negative achieved judgement about a node.

In summary, our approach is to exploit the reputation about a node A, which is actually distributed among A's neighbours in the network in order to achieve an average value for this reputation to be suggested to a new node when he has to assign a trust level to A. Our reputation-based system is used to evaluate trust, hence to modify trusting network over time, also exploiting feedback mechanisms.

The paper is organized as follows: in section 2 we describe how a person interacts with an existing trust-based (i.e. friendship) community with whom he is initially unacquainted, formalizing our discussions into a trusting algorithm; trust links lifecycle are introduced in section 3, whereas in sections 4 preliminary results of network simulation are shown. Finally, in sections 5 and 6 we respectively present related works and our conclusions.

2. The mechanism of trusting within social networks.

2.1. Concepts and Terms.

According to the definition given in [13], we say a person (Alice) trusts another one (Bob) if A is persuaded that information coming from B are true, for example, if A needs B to help her in finding a job, and B says that he will help A, A is guaranteed that B says the truth, hence he will actually help A. This definition expresses the same concepts introduced in [4], where trust is intended as the choice A makes when she perceives an ambiguous path and B's actions determine how the result of following the path will be; a similar definition is given in [17], where "trust is a bet about the future contingent actions of others". We quantify how A trusts B with a number $t_B^A \in [-1, 1]$, where -1 means A does not trust B at

[*] Dipartimento di Ingegneria Informatica e delle Telecomunicazioni, Facoltà di Ingegneria, Università degli Studi di Catania – Viale Andrea Doria 6, I95125, Catania, ITALY – {car,alongheu,mm,gmangioni,vnicosia}@diit.unict.it

all, and 1 indicates that A trusts B completely; a value of 0 models the *indifference*, due to either a lack of information about B or when exactly opposite judgements come from the network, hence A is unable to decide whether trust or not in B.

We will refer to trust value also as a "risk factor", i.e. the risk A accepts when she believes that B's actions will lead to a good outcome. After introducing a definition for trust, we have to consider that trust is strictly related to acquiring information, i.e. in a social network a trust relationship between A and B is established if A collects some information about B, either with a direct interaction with B or even getting such information from other sources, e.g. his acquaintances, web sites, e-mails, contact board, and so on; indeed, if A does not know anything about the existence of B, A also cannot assign any trust value to B. We name these information about B as B's "resume", representing information as study degree, working experiences, personal skills and attitudes, and so on; at this stage, we do not impose any constraint on resume's content or structure

We denote the resume about a node X owned by a node Y as:

$$C_X^Y = \{\mathsf{X}, \overline{C_X^Y}\} \tag{2.1}$$

where X is the identifier for the node X the resume is about, and $\overline{C_X^Y}$ is the set of information cited previously. Note that different nodes actually can know different things about X, as in real world occur, thus we indicate the owner node Y; moreover, resumes evolve over time, as soon as Y acquires information about X, thus updating $\overline{C_X^Y}$. The most updated and complete resume is the one owned by the person it concerns, simply indicated as $\overline{C_X}$.

2.2. A sample scenario. To model trusting mechanisms inside a community, we use a directed labeled graph $\mathcal{G}$, defined as a pair $(\mathcal{N}, \mathcal{E})$ where $\mathcal{N}$ is a set of nodes and $\mathcal{E}$ is a set of edges (or links) between vertices, i.e. $\mathcal{E} = \{(u, v) | u, v \in \mathcal{N} \wedge u \neq v\}$; a link from A to B is labelled with the pair trust, resume, i.e. (t_B^A, C_B^A)

To show how links are established, let us consider Alice (A), an unacquainted person who wants to establish some contact with an existing community, usually to satisfy some needs in her everyday life. For instance A may be a new student that needs to know where classrooms are located, hence she asks for some help to other students she meets in a corridor, or she can search for a computer science professor on the University website. In the first case, A directly contacts other individuals, say Bob (B) and Carl (C), thus having links form A to B and from A to C, whereas in the last case A collects information without a direct interaction with individuals, but she is still collecting resumes about some professors (say Danny and Eve, respectively node D and E), so links from A to D and E are present too. Note that arcs are oriented since mutual trust values are not necessarily the same, or one of them could also be not present, for instance A could completely trust in B and C, since she is in a new, unknown context, but B and C might assign a low trust value to A, being an acquaintance they are meeting for the first time; moreover, A will assign trust values to professor D and E based on their resumes, but D and E actually do not know anything about the existence of A, hence neither arc from D to A nor from E to A will be present.

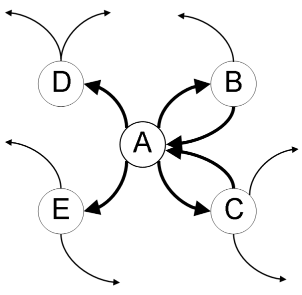


FIG. 2.1. *Node A joining the network*

In this scenario, represented in figure 2.1, where thinner lines represent trusting relationships with other nodes, A assigns a first trust value to nodes B,C,D and E, that will be refined over time, as soon as other information (resumes) is available

In this scenario, A assigns a first trust value to nodes B,C,D and E. Such values will be refined over time, as soon as other information (resumes) is available, e.g. A can re-evaluate t_D^A and t_E^A as soon as she meets those professors; similarly occurs for trust values assigned to A by nodes B and C.

2.3. Trust evaluation. In addition to direct (personal) interaction between two individuals, in the real world the refinement process of a trust value assigned by a person A to a generic person X, not belonging to the set of A's acquaintances, is often performed by simply asking to X's acquaintances, i.e. to those people who directly know X and can provide a judgement about him. These judgements, together with all resumes about X, are collected and mediated by A, so she can refine her knowledge and assign a proper trust value about X exploiting other people's personal experience with X. Note that in the graph model, the term "acquaintance" correspond to a node that is neighbour to another, i.e. a path of length 1 (an arc) exists between these nodes.

In this paper we focus on the trust refinement mechanism exploiting acquaintances opinions, since it better represents social network interactions. A, before asking information about X, setup the net by adding X, if not present, and creating an arc pointing to it labelled by $risk_A$, that represents a default value typical of A.

```
addToNet
Require: $\mathcal{G}$ is the graph representing the net
Require: $A, X \in \mathcal{N}$
Require: $risk_A$ is A's ingenuity
1: create a new arc in $\mathcal{N}$ from A to X: $arc(A, X)$
2: label $arc(A, X)$ using $\{risk_A, C_A^X\}$
```

The *trustNode* algorithm implements the evaluation of mediated trust performed by a generic node A about a node X; in the following we also comment relevant instructions.

```
trustNode
Require: $\mathcal{G}$
Require: $A, X \in \mathcal{N}$ where A is a node that wishes to trust node X
Require: $C_X$
Require: $\tau_{good}$ that is threshold used to select node for query forwarding
Require: $n_r$ maximum number of request, $id_q$ is a query identifier
1: if $X \notin \mathbf{N}^A$ then
2:    $addToNet(A,X)$
3: end if
4: $\overline{\mathbf{R}} = \{r_i \in \mathbf{R}_X^A | r_i > \tau_{good}\}$
5: $\mathbf{T}_X^A = \emptyset$
6: for all $(I \in \overline{\mathbf{R}}) \wedge (| \mathbf{T}_X^A | < n_r)$ do
7:    $t_I^X = forward(I, C_X, id_q)$
8:    $\mathbf{T}_X^A = \mathbf{T}_X^A \bigcup \{I, t_X^I\}$
9: end for
10: if $| \mathbf{T}_X^A | < n_r$ then
11:    for all $(I \in (\mathbf{N}^A - \overline{\mathbf{R}})) \wedge (| \mathbf{T}_X^A | < n_r)$ do
12:       $t_X^I = forward(I, C_X, A, id_q)$
13:       $\mathbf{T}_X^A = \mathbf{T}_X^A \bigcup \{I, t_X^I\}$
14:    end for
15: end if
16: $t_X^A = trust(\mathbf{T}_X^A, \mathbf{N}^A)$
```

First (line 1-3), A check whether X does not already belong to the set of her acquaintances (denoted as $\mathbf{N}^A$); in this case the *addToNet* function creates the arc from A to X.

Now, A would collect opinions about X to X's acquaintances, but probably she does not know anyone of them, hence A asks *her* personal acquaintances whether they know X, or whether acquaintances of A's acquaintances know X and so on, actually forwarding her request (*query*) through the network by exploiting trusting relationships. Specifically, A initially establish how many opinions (say, n_r) about X she needs to assign X a trust value: the more opinions will ask, the more important for A the evaluation is, the more accurate the trust value to assign will be; the drawback will be the flooding of messages traversing the network. To model real world interactions, A establishes an order according to acquaintances are contacted. In particular, A first considers acquaintances she trusts more, at the same time having a resume similar to X's resume, to show why we consider resumes similarity, suppose for instance that from X's resume, A knows that X studies nuclear physics at the university, and A also suppose that trusts B and C the same, but B also is a nuclear physics student. To assign a trust value to X, in the real world A will first ask B, since A supposes that similarity in resumes increases the possibility that those acquaintances directly know X.

To formalize this discussion, we introduce the *correspondence* function to evaluate resumes similarity and *relevance* function to sort acquaintances based on correspondence and assigned trust level:

- *correspondence* is defined as $corr : C \times C \to [0;1]$, also noted with $corr(C_I, C_X)$; we call $c_{I,X}$ the result of correspondence between nodes I and X. At this stage, the implementation of this function simply adopt classical information retrieval techniques aiming at extracting inverse index vector for both resumes. Obviously, $corr(C_I, C_X) = corr(\overline{C_I}, \overline{C_X})$.
- *relevance* function is defined as $r_I^{A,X} = c_{X,I} \cdot t_I^A$ where A is the node that ask an opinion about X to one of her neighbour I; the set of all values is:

$$\mathbf{R}_X^A = \{r_I^{A,X} | \mathsf{I} \in (\mathbf{N}^A - \{\mathsf{X}\})\} \tag{2.2}$$

Given these definitions, in line 4 of the algorithm A establishes a threshold τ_{good} to find acquaintances she both trusts a lot and also with a resume highly similar to X's. These *qualified* acquaintances are considered first (lines 6-9) when A sends the query (line 7), using the $forward$ function to collect their opinions about X, finally storing this trust value together with the neighbour that provided it in the following set (line 8):

$$\mathbf{T}_{\mathsf{X}}^A = \{(\mathsf{I}, t_X^A) | \mathsf{I} \in \overline{\mathbf{N}}_A\} \tag{2.3}$$

$\overline{\mathbf{N}}_A$ indicates the set of acquaintances (neighbours) who actually answered the query. Note that acquaintances with high relevance are randomly selected when sending queries in order to avoid always contacting the same acquaintances.

A starts sending queries to qualified acquaintances, aiming at getting n_r answers from them, but when their number is not enough or when the number of successfully queries is not enough (line 10), A will continue sending queries to less qualified acquaintances (lines 10-15); note that A might not get all n_r answers, as in real world occurs.

Finally, the collected answers are used to build a mediated opinion about X. When evaluating this average trust level opinions coming from acquaintances should be weighted according to trust level A assigned to them, reflecting the real world behaviour. The formula we adopt is the following:

$$t_X^A = trust(\mathbf{T}_X^A, \overline{\mathbf{N}}_A) = \frac{\sum t_X^i \cdot t_i^A}{\sum t_i^A} \tag{2.4}$$

This is essentially a weighted average formula where main terms are t_X^i, i.e. acquaintances opinions about X, weights are t_i^A, that is trust values assigned by A to his acquaintances. To understand why t_i^A is also present at the denominator, we impose that acquaintances A assigned low trust values should have less influence over the resulting mean value, in order to reflect the real world behaviour adopted by a person, i.e. he tends to neglect the opinion of an acquaintances he trusts in with a low degree. To do this, at the denominator we place the sum of *contributions*, being each acquaintance's contribution the trust value A assigned him; if A trusts with a low degree someone (i.e. related trust value tends to 0), his contribution as well as also the corresponding term at the numerator will be neglected. The t_X^A is finally associated to the arc from A to X.

A relevant question is that when each of A's acquaintances receives the query, his task is to answer the query with his opinion about X, but the current acquaintance could still get into the same trouble, i.e. he does not know directly X; similarly to A, such node will forward the request to his acquaintances (but A), waiting for a set of trust values about X to be mediated as described previously, hence the resulting mean will be assigned by the node to X and it will be also delivered to A (or generally to the requesting neighbour node). The propagation of the query will occur until one of X's acquaintances is reached. The $forward$ function uses an algorithm (not reported here) very similar to the $trustNode$, except for two mechanisms:

1. Similarly to the first node that generated the query (A), an *intermediate* node needs n_r opinions about X, but differently from A, if n_r is not achieved exploiting his qualified acquaintances, no more answers will be asked to acquaintances having a relevance below the threshold, modeling the absence of a personal interest the intermediate node has in determining an opinion about X. In the worst case, the intermediate node will give back a trust value of 0 (i.e. he will answer he cannot give a significant opinion about X). The n_r and τ_{good} are the same A established; this models the propagation of the *importance* A assigned to the query.
2. The formula adopted by each intermediate node to determine the mediated trust value is the same adopted by A, but the trust value the intermediate node returns is multiplied by a distance factor, i.e. a value in the range]0,1] that models the fact that in the real world we consider more significant an opinion if it comes from a person that directly knows X, decreasing the value of this opinion as soon as increases the number of individuals to be contacted to get to this opinion.

3. Trusting relationships lifecycle. Social networks dinamically evolves by changing both their arcs (i.e. trust values) and nodes (coming and leaving the trusting network), so topology between requests changes. In the following we describe such situations.

3.1. Feedback. Acting as in social networks, whenever a node B behaves positively (good actions), any node A who assigned a trust level to B should raise her reputation about B, lowering trust in the case of B perpetrates some bad action, hence any trust value about someone should be always subject to changes whenever we get information about that person. Note that if this does not occur, this would denote that A is not capable of processing external information, as it happens for some diseases involving brain damages; however, we do not focus on these situations, hence we suppose that any information about individuals coming form the external world is processed and leads to re-evaluations of associated trust levels.

Re-evaluation of trust about B could also be triggered by A itself if the trust level was evaluated a long time ago; we name *aging* this situation (see next paragraph for more details).

No matter which is the reason, whenever a node A evaluates or re-evaluates a reputation about a node B and consequently adjust B's trust level, according to the algorithm illustrated in section 2, A should also spread this new information over the network, in order to inform about B's behaviour (possibly different from the past); we call *feedback* this mechanism. Usually A cannot inform the entire network, rather he/she propagates the information to his neighbours (but B), allowing them to adjust their opinion about B. The set of neighbours considered is $\overline{\mathbf{N}}_A$, i.e. the set of acquaintances (neighbours) who answered the re-evaluation query. Among these neighbours, A considers the subset of those satisfying the following conditions:

1. their trust value is over a given threshold, i.e. A strongly trusts them,
2. the trust level about B they gave to A in the past is very different from the reputation evaluated by A; note that A got the vector of neighbours' judgements about B during the re-evalution process. itself

Both conditions aim at modelling the behaviour within social networks, in particular A considers strongly trusted neighbours *worth* to be adviced about her trust level about B (first condition), and inform them only if this is useful (second condition). Also note that A could extends this feedback also to other neighbours, e.g. those belonging to $\overline{\mathbf{N}}_A$ sets used in past (not the last) B reputation evaluation process; this depends on how much *memory* about past evaluation A retains; in our approach, we consider just the last one.

Another issue to address concerns the behaviour of one of A's acquaintances (say C) as soon as she receives the new trust value about B, for instance if C receives a strong negative feedback from A about B and C assigned a high trust level to B and a lower to A, should C decrease his trust on B (i.e. should he believe to A's judgement?) Or should he decrease his judgement about A' (since she provided an unaffordable opinion)? Or should C decrease both?

Our choice is always inspired by real world behaviour, hence as soon as node C receives B's trust level, he first consider his trust level about A; if this is below a given threshold, he will not consider information coming from A to modify his opinion about B neither perform any propagation. If C strongly trusts A, he:

1. uses A's trust value about B to re-evaluate his trust level about B; note that this does not trigger the algorithm introduced in section 2, rather C simply introduces A's judgement about B into the formula 2.4, substituting previous A's contribution if A was already included in $\overline{\mathbf{N}}_A$, or properly extending $\overline{\mathbf{N}}_A$ otherwise.
2. once a new updated judgment about B has been evaluated, C can stop or he can further propagate this information to a subset of his neighbours, according to the same criterion adopted by A previously.

We choose to propagate, so the information can be actually spread over the trust network; note that in order to avoid excessive network traffic, we adopt a simple solution similar to the *distance factor* introduced in the previous section, i.e. we consider the inverse of hops number (from A) involved in the propagation process, and a node along this chain will stop only when this factor will be lower than a given threshold.

3. C does not change his opinion about A since in this case he trusts her enough to consider her judgement about B as valid; in other words, it would be inconsistent a modification over C's opinion about A.

Feedback mechanism we introduced heavily influences trusts, infact modifing trust values too often could make the net unstable whereas infrequent modifications would not permit an accurate perception of "vox populi" about nodes. In summary, the feedback mechanism reflects the real world behaviour that occurs when A evaluates a more recent opinion about a person and wishes to communicate this opinion to all of her trusted acquaintances, since she thinks that this opinion is more affordable than other existing opinions her acquaintances currently own. Further works should be devoted to a deep analysis of threshold influences the feedback mechanism.

3.2. Weak and Strong links. Each time a node A collects information about an unacquaintance node X and consequently assigns X an initial trust value without involving his acquaintances, we name the arc from A to X as *weak* link, since the value of trust has not been verified neither by A's personal experience, nor by collecting acquaintances opinions; as soon as A has a personal experience with X or she gets other opinions and build a refined trust value, the arc will be named as *strong*. A weak link is used to model the first contact a person has with one another, a contact that implies the assignment of a first, temporary and possibly wrong trust level, as in real world actually occurs; for instance, all links A creates with nodes B,C,D and E in section 2 are initially weak.

A strong link can be lowered to a weak one when its associated trust level has been evaluated a very long time ago, hence that link should be not considered affordable anymore, and a new evaluation should be performed. Similarly, if a weak link does not become strong for a long period, it will be removed from the network, modeling the fact that A forgets about X existence if he never contact X during his life. This *aging* mechanism is also useful to avoid a network with an excessive (and probably useless) links.

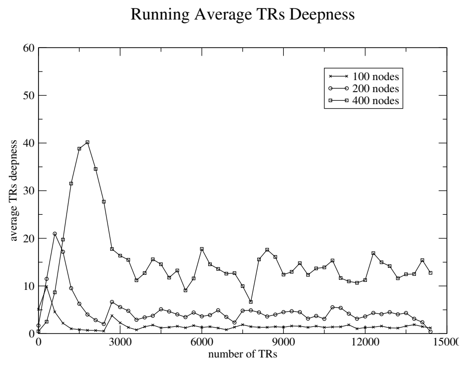
Finally, note that the mechanism of evaluating trusting by collecting opinions from acquaintances relies both on weak and strong links.

4. Preliminary results. A first set of simulation has been performed using the trusting algorithm described previously, in addition with aging mechanism, in order to test the convergence of the network. Simulation have been performed considering networks with different number of nodes (i.e. 100, 200 and 400), with initially no links; the trusting algorithm is invoked when queries are generated, and it allows the growing of links number, hence the network evolves through a transient state until a stable set of connections is achieved. Input parameters we assigned for trusting algorithm are $n_r = 3$ (in order to avoid the flooding of the network), and distance factor=0.9 (to allow long paths for query forwarding to be considered). Results are quantified by introducing two measures:

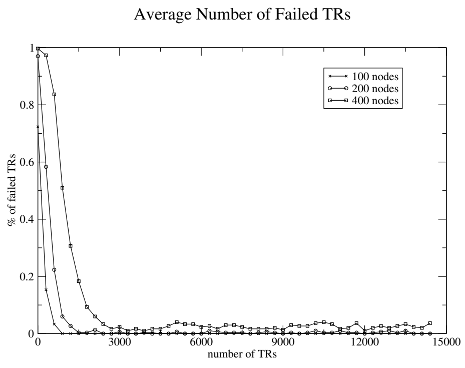
1. average trusting requests deepness, denoted as $ATRD$ (where a TR is a query), indicating the deepest path from the requesting node to the node to be trusted, mediated among all TRs; this property is used to express the *efficiency* of the trusting process
2. average number of failed TRs ($AFTR$), where a TR is considered *failed* when no answers are collected, e.g. when no relevant nodes are found for TR forwarding. This property is used to express the *effectiveness* of the trusting process

In fig. 4.1(a) and 4.1(b) running $ATRD$ and $AFTR$ are represented, respectively. These express how many TR are needed for the network to converge, i.e. to provide stable values for both properties. This two experiments clearly point out that the proposed trusting algorithm converge and, in particular, TRs deepness reachs stable values when about 1500–3000 (for networks of 100 to 400 nodes, respectively) TRs are performed. Note that TRs deepness initially increases since when the network is created too few links are present; as soon as links number increases, the deepness decreases since more (possibly shorter) paths are available. The number of failed TRs rapidly decrease to zero when the network converges, meaning that when a trusting request is issued, exactly n_r opinions are received from the source node allowing it trust evaluation according to eq. 2.4.

In addition to $ATRD$ and $AFTR$ in this paper we shortly discuss the algorithm resilience to group empowering. This measure shows which is the critical cardinality of a cheating group to be able to raise (or equivalently



(a) Average TRs deepness



(b) Average number of failed TRs

Fig. 4.1. *Network evolution*

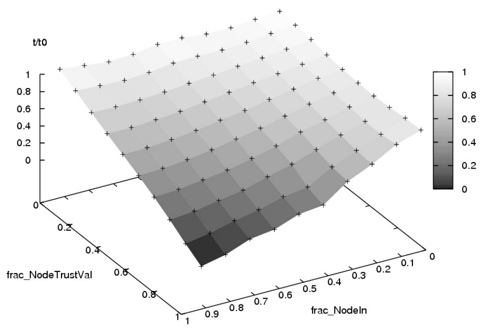


Fig. 4.2. *Resilience to group empowering*

lower) the reputation of a target peer. The experiment is conduct on a network of 100 nodes where 15000 TRs are performed (as previously discussed, in this case the network reaches a stable state). Then, we randomly choose a source node S, a target node T and a TR from S to T is executed; the corresponding trust level $t0_T^S$ is used as a comparison factor during the subsequent TRs. This experiment is repeated lowering from 0% to 100% the trust value ($frac\_NodeTrustVal$ in figure 4.2) of a given fraction ($frac\_NodeIn$ in figure 4.2) of the number of nodes having T as neighbour (n ranges from 1 to $|\mathbf{P}^{\mathsf{T}}|$, $\mathbf{P}^{\mathsf{T}} = \{\mathsf{U} \in \mathcal{N} | (\mathsf{U},\mathsf{T}) \in \mathcal{E}\}$).

For a given pair ($frac\_NodeTrustVal$, $frac\_NodeIn$), we evaluate t_T^S hence the ratio $t_T^S/t0_T^S$. The experiment is iterated changing source and target nodes in order to obtain the average real network behaviour. The ratio gives an idea about the algorithm robustness against group empowering; indeed, as shown in figure 4.2, a group of cheating nodes can heavly affect the judgement on a target node (i.e. lowering $t_T^S/t0_T^S$) only when $frac\_NodeIn$ is a significant fraction (about 40-50%) over $|\mathbf{P}^{\mathsf{T}}|$, or when $frac\_NodeTrustVal$ grows up to about 40%.

5. Related work. Trust has been studied in social sciences, business and psychology before it became central to computer science research; the survey [1] offers a complete overview on most recent issues concerning trust.

In [10], the first work providing a formal model of trust, the value of trust is chosen within the range [-1,1], covering from complete distrust to full trust. We follow the same approach, since we believe this is the best representation: simple, normalized and symmetric around the zero (indifference). As in [10] is claimed, probably none of the extremes (i.e. full trust or distrust) is actually possible. In [16] however, the use of a range with negative values to model distrust is somehow criticized, mainly due to algorithmic-related issues (no more necessarly real values for trust matrix eigenvector); authors also debate about whether a trust score of 0 translate to distrust or to *no opinion*; we use the [-1,1] range since negative values provide the right adjustment when evaluating reputation of a given node.

The paper [11] shows some similarities with our work. First, authors also view the question of trust in the computer science context as a propagation of social matter ("in today's connected world, it is possible and common to interact with unknown people ..."). The metric they define for trust evaluation is based on their previous work, MoleTrust, that predicts the trust score of source user on target user by walking the social network starting from the source user and by propagating trust along trust edges; intuitively the trust score of a user depends on the trust statements of other users on her and their trust scores. This approach is quite similar to our proposal, except for some aspects:

- to stop walking on the network, authors define the *trust propagation horizon*, which specifies the maximum distance from source user to which trust is propagated along trust chains. We instead claim that our *distance factor* is a better choice since it gracefully decreases the importance of nodes along the walk, whereas with the horizon the walk is interrupted.
- for predicting the trust score of a user, MoleTrust analyzes the incoming trust edges and discards the ones coming from users with a predicted trust score less than 0.6 *threshold*; our approach allows a finer granularity in deciding trusted nodes by exploiting the *relevance*.
- both approaches share the difficulty of applying the local trust metrics to real available data set as Epinions.com, since they often accept only binary trust values instead of continuous (but more realistic) values.
- Finally, in the MoleTrust-based approach, it is deeply analyzed the role of controversial users, i.e. those who are judged in very different ways by other nodes. In particular, they show that a local trust metric (i.e. trust assigned to a node depends on the point of view of the evaluating node) is better than a global one since it significantly reduces the prediction error for controversial users. Our approach put both positive and negative opinions in the same formula when evaluating reputation, hence it is not required to distinguish opinions.

In the last decades, the recommender system is gaining an important role in today's networked worlds because they provide tool to support decision helping in selecting reliable from unreliable. Reliability is often expressed through a trust value with which each agent labels its neighbours; [14, 9] explore this, but they does not investigate in the topic of formation of trust based on real-world social studies. Some recent works have suggested to combine distributed recommendation systems with trust and reputation mechanisms [14, 12].

People surfing the Web has already faced the matter to form opinion and rate them against trust, as far as the well-known reviewers' community *Epinions (*`http://www.epinions.com`*)* which collects reviews from the community members, any members can also decide to "trust" or "distrust" each other. All the trust and block relationships interact and form a hierarchy known as the Web of Trust. This Web of Trust (WOT) is combined with rating to determine in what order opinions are shown to readers. However trusting model remains centralized (trust is influenced only by the manager).

Ziegler and Golbeck [6, 18] believe that computational trust model bear several favorable properties from social filtering than they expect notions of trust must reflect user similarity. Therefore a reputation system is an important tool in every network, but assume a central role in emerging P2P networks where many people interacts with many others in a sort of "democratic" fashion. Some author discusses decentralized methods that approximate ranks according to local history and a notion of neighbourhood [2] where trust is calculated trying advantage of small-world properties often emerging in networks that mimic real world. In P2P area EigenTrust [8] propose a distributed implementation of PageRank [15] that needs also a distributed structure to store data and imposes to pre-trust all nodes belonging to the net thus reducing the "de-centralisation". The forward algorithm introduced in previous section proposes a mechanism for trusting propagation; a little taxonomy of other possibile propagation criteria can be found in [16]. Our approach is someway similar to their *direct propagation* mechanism; we believe others mechanisms described seem to be a limited validity.

In [18], they expect notions of trust to clearly reflect user similarity; this is similar to our *relevance* and *correspondence* factors, whose aim is indeed to take into account the similarity of resumes.

6. Conclusions and future work. This paper introduced an approach to trust entities hint at human behaviour; we proposed an algorithm based on local behaviour of an human and we believe that several important properties of social network will emerge. The algorithm has been discussed and motivated thruoghout the paper; preliminary promising results about the convergence of the network and its resilience to group empowering have been shown.

Further studies are currently active on testing network behaviour, as well as on investigation about communities that may emerge—of interest, similarity, confidence, etc—and on the importance of *correlation* among resumes to improve the effectiveness and efficiency of the proposed approach.

REFERENCES

[1] D. Artz and Y. Gil, *A survey of trust in computer science and the semantic web*, Web Semantics: Science, Services and Agents on the World Wide Web, 5 (2007), pp. 58–71.

[2] M. Dell'Amico, *Neighbourhood maps: Decentralised ranking in small-world p2p networks*, in 3rd International Workshop on Hot Topics in Peer-to-Peer Systems (Hot-P2P), Rhodes Island, Greece, April 2006.

[3] Z. Despotovic and K. Aberer, *P2P reputation management: probabilistic estimation vs. social networks*, Comput. Networks, 50 (2006), pp. 485–500.

[4] M. Deutsch, *Cooperation and trust, some theoretical notes*, in Nebraska Symposium on MOtivation, N. U. Press, ed., 1962.

[5] J. Golbeck, *Trust and nuanced profile similarity in online social networks.* ACM Transactions on the Web, to appear.

[6] J. Golbeck and J. Hendler, *Reputation network analysis for email filtering*, in Conference on Email and Anti-Spam (CEAS), Mountain View, CA, USA, July 2004.

[7] M. Gupta, P. Judge, and M. Ammar, *A reputation system for peer-to-peer networks*, (2003), pp. 144–152.

[8] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, *The eigentrust algorithm for reputation management in P2P networks*, in In Proceedings of the Twelfth International World Wide Web Conference, 2003.

[9] M. Kinateder and S. Pearson, *A Privacy-Enhanced Peer-to-Peer Reputation System*, in Proceedings of the 4th International Conference on Electronic Commerce and Web Technologies (EC-Web 2003), K. Bauknecht, A. M. Tjoa, and G. Quirchmayr, eds., vol. 2738 of LNCS, Prague, Czech Republic, September 2003, Springer-Verlag, pp. 206–215.

[10] S. Marsh, *Formalising trust as a computational concept*, tech. report, Department of Mathematics and Computer Science, University of Stirling, 1994.

[11] P. Massa, P. Avesani, *Controversial users demand local trust metrics: An experimental study on epinions.com community.*, in AAAI, 2005, pp. 121–126.

[12] P. Massa, B. Bhattacharjee, *Using trust in recommender systems: An experimental analysis*, in iTrust, 2004, pp. 221–235.

[13] F. C. Mish, ed., *Dictionary and Thesaurus*, Merriam-Webster, Incorporated, 2007.

[14] M. Montaner, B. Lopez and J. L. de la Rosa, *Opinion-based filtering through trust*, in CIA '02: Proceedings of the 6th International Workshop on Cooperative Information Agents VI, London, UK, 2002, Springer-Verlag, pp. 164–178.

[15] L. Page, S. Brin, R. Motwani, and T. Winograd, *The pagerank citation ranking: Bringing order to the web*, tech. report, Stanford Digital Library Technologies Project, 1998.

[16] P. Raghavan, R. Guha, R. Kumar, and A. Tomkins, *Propagation of trust and distrust*, in Proc. of WWW2004 conf., 2004.

[17] P. Sztompka, *Trust: A sociological theory*, (1999).

[18] C.-N. Ziegler and J. Golbeck, *Investigating correlations of trust and interest similarity—do birds of a feather really flock together?*, Decision Support System, (2005).

Edited by: Maria Ganzha, Marcin Paprzycki
Received: Jan 31, 2008
Accepted: Feb 9, 2008

Scalable Computing: Practice and Experience
Volume 9, Number 1, pp. 39–49. http://www.scpe.org

ISSN 1895-1767
© 2008 SCPE

APP: AGENT PLANNING PACKAGE[*]

SAŠA TOŠIĆ[†], MILOŠ RADOVANOVIĆ[†], AND MIRJANA IVANOVIĆ[†]

Abstract. Nowadays, a large number of planning systems exist. The majority of them are either applicable only to specific domains (domain dependent) or support only plan generation, but not plan execution and supervision. This paper presents Agent Planning Package (APP), a new domain independent planner which facilitates the creation of intelligent agents. Through the use of APP, agents can generate plans within their operating environment, as well as execute them and supervise the execution. Furthermore, APP supports online changes to the planning problem and domain, as well as creation of multiple plans and plans which contain alternative paths of execution.

1. Introduction. Modern mainstream application development tools offer many features which simplify and quicken the development process. Developers can easily create and incorporate databases into their projects, design user interfaces, enable network communication of programs, use extensive libraries which implement standard data structures and algorithms, etc. Development tools also support the break-down of larger problems into smaller ones, and merging of small problems into larger, but these operations have to be performed manually by the programmer. None of the standard development tools offers *automatic* combining of small functional parts, in the sense of the tool itself generating their order of execution to solve a larger problem. Furthermore, the tools provide little control of pre- and post-conditions for the execution of a part of a program, except for debugging purposes, which again requires the problem solution to be specified by hand. The craft of manual problem solving is one of the most demanding and error-prone parts of the application development cycle.

On the other hand, *planning systems* solve complex problems by combining atomic actions which can be performed. They use a declarative description of actions and the goal, based on which they create a plan, i.e. a sequence of actions which need to be executed in order to take the system from the initial state to the state with the desired properties.

When specifying actions one needs to specify their parameters, the preconditions which must hold before the action can be executed, and the effects of the actions. A large number of planning systems also support specifying of the cost of action execution, and the probabilities that actions will be performed successfully.

Although the integration of concepts from planning systems into mainstream application development tools seems far away, the coupling of automatic planning with *agent development systems* is a reality. This paper presents Agent Planning Package (APP), a new domain independent planner which facilitates the creation of intelligent agents. Through use of APP, agents can generate plans within their operating environment, as well as execute them and supervise the execution. Furthermore, APP supports online changes to the planning problem and domain, as well as creation of multiple plans and plans which contain alternative paths of execution. The main purpose of APP is not to present a new deliberative agent architecture (although it may be possible to treat it as such), but rather to enable the extension of existing architectures with planning capability.

The rest of the paper is organized as follows. Related work and major existing planning systems are reviewed in Section 2. APP is introduced in Section 3, through a discussion of its basic anatomy and the plan creation process. Section 4 introduces a simple logistic domain as a working example that will be used to illustrate APP's features throughout the paper. Section 5 describes the APP planning cycle in more detail, focusing on the agents' needs and point of view. Several possibilities for implementation of a logistic agent which utilizes APP are given in Section 6. Section 7 summarizes experimental results which show that APP is among the medium-powered planners in terms of speed, and among the most efficient in terms of generating the shortest existing plan if one exists. Finally, Section 8 concludes and outlines the directions for future work.

2. Related Work. Research on planning systems (more precisely, automated problem solving) started in the late 1950s. One of the first programs in this field was GPS (General Problem Solver) by Newell, Shaw and Simon [18]. In 1971, Fikes and Nillson developed the STRIPS (STanford Research Institute Problem Solver) formalism [8], which was the basis for most research in the field until Pednault presented ADL (Action Description Language), mixing the STRIPS approach with the situation calculus [19]. The ADL formalism provided inspiration to many researchers, and soon other formalisms like UCPOP and UMCP emerged. In 1998, PDDL (Planning Domain Definition Language) was introduced [15], attempting to unify the most important aspects of all prior formalisms. PDDL 2.1 added the possibility to specify durations of actions, and integrated the systems for planning and scheduling [9]. In 2002, this version was

[*]This work was supported by the Serbian Ministry of Science and Environmental Protection.

[†]University of Novi Sad, Faculty of Science, Department of Mathematics and Informatics, Trg D. Obradovića 4, 21000 Novi Sad, Serbia, {sashat,radacha,mira}@im.ns.ac.yu.

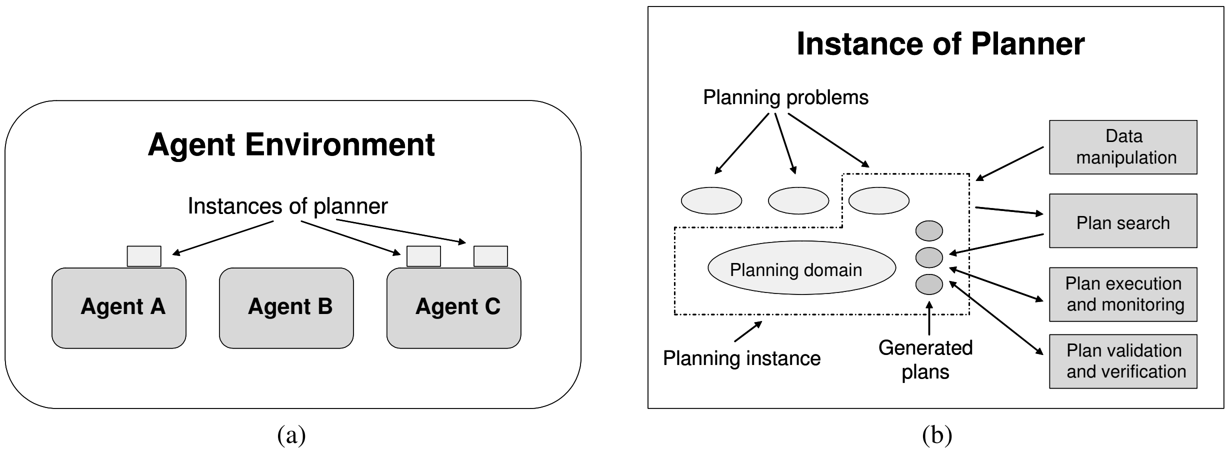


FIG. 3.1. *Structure of a multi-agent environment in which agents utilize APP*

further extended to PDDL 2.2 by introducing derived predicates and timed initial literals [7]. Year 2005 witnessed the introduction of PDDL 3.0, which enabled the definition of strong and soft constraints on plan trajectories, as well as strong and soft problem goals [10].

In parallel with the development of planning formalisms, planning systems based on them were being written. Some of the well known systems include SHOP [17, 16], GraphPlan [4], STAN [14], FF [12], HSP [5], SAPA [6] and AltAlt [20].

3. The Agent Planning Package. APP is a planning system written in Java in form of a Java package. Java was chosen as the implementation language for a number of reasons, including its portability and the existence of numerous Java-based agent systems, e.g. JADE [3], Cougaar [11], AJA [2], etc.

The basic purpose of APP is to facilitate the creation of intelligent agents through integration with existing agent development tools. Thus, APP is designed to be simple to use and to offer vast functionality to an agent, such as the creation of planning instances, their modification, creation of different types of plans and their execution in the agent's environment.

3.1. The Basic Anatomy of APP. The structure of a multi-agent system in which agents utilize APP is shown in Fig. 3.1(a). Each agent in the environment may use an arbitrary number of instances of the *planner*, according to its needs. As illustrated in Fig. 3.1(b), one instance of the planner may be used to describe and initiate several *planning instances* which use the same *planning domain*, but may differ in the concrete *planning problem*.

The planning problem describes the objects existing in the environment, declares initial values of functions, defines the relations that exist in the environment and describes the goal. This approach insures that there is no duplication of data characteristic of the planning domain, e.g. definitions of types, predicates, functions, constants and actions.

The core of APP resides in class Planner, which forms the base class for creating instances of the planner. At the instance creation phase, a reference to an agent is passed to the planner; this reference is used at the plan execution stage.

The class Planner contains the field domain, which represents the planning domain, and a list of planning instances. Creation of planning instances is done via a call to the addInstance() method, which returns the planning instance's ordinal number. The instance itself is accessible through method getInstance(num). Besides creating planning instances, it is possible to delete them, get the number of instances and the ordinal number of the last instance, and delete the planning domain if there are no planning instances left.

3.2. Plan Creation in APP. Plan creation is done on the basis of the current state of the planning instance prior to the invocation of one of the plan creation methods. The used algorithm is a mixture of several standard algorithms which utilize a planning graph. The base algorithm is the one used in the GraphPlan system, and is described in [4], while some of the employed modifications stem from [9, 12, 13].

During the creation of the planning graph, every level of the graph is represented by two sublevels, one representing facts (instances of predicates), and fluents (instances of functions), and the other representing instances of actions. The zeroth level of the graph is initialized with facts and fluents from the internal data structure which represents the planning problem[1]. After the creation of every level of a planning graph that may contain a plan, a backward-chaining strategy

[1]The zeroth level of the graph does not contain any action instances.

is used to find the plan. If this search fails, the next level of the graph is created. The algorithm used in the current implementation is outlined in Fig. 3.2.

```
Function createPlan
    0: Create zeroth level of graph
    1: While possible to find a plan and not created all goal facts do
          Create new level of graph
    2: While possible to find a plan and there exist goal facts in mutex do
          Create new level of graph
    3: While possible to find a plan and plan not found do
          Call Backward-chaining strategy
          If plan not found then
             Create new level of graph
    4: If plan found then return plan
                     else return null
```

FIG. 3.2. *Plan creation algorithm used in the current implementation of APP*

Steps 0 and 1 of APP's plan search algorithm involve successively creating levels of the graph until a level is reached which contains all the facts specified in the goal. In step 2, the creation of new levels is resumed, until a level is generated which contains no mutually exclusive facts. If any of the two terminating levels from steps 1 or 2 is determined not to be reachable (i.e. it is not possible to find a plan), the algorithm halts.

The search for the plan itself is conducted in step 3, using a backward-chaining strategy. First, all possible ways of fulfilling the goal are instantiated at the highest (goal) level. This generates all subgoals whose fulfillment on the previous level can lead to the fulfillment of the goal, and also determines the sequences of actions which transfer the system from the subgoal to the goal. The subgoals are then sorted according to a heuristic[2], and their fulfillment is respectively attempted at the previous level. If the root level of the graph is reached, a plan is generated according to the actions chosen during the search. If the root level is unreachable, the subgoal is inserted into a goal tree data structure for its level. During subsequent descents into lower levels of the graph, new subgoals are tested for presence in the goal tree, and if the outcome is positive the search is aborted for that particular subgoal.

The goal tree is a data structure in APP which is filled with goals that were not satisfied at a particular level of the graph. Since every goal is represented by a (sorted) list of numbers denoting facts, the goal tree is implemented as a *multiway trie*, more precisely an *existence trie*, described in [21] (Chapter 15). Each node of the tree contains a single fact, and every path from root to a leaf represents a goal. At goal insertion, the first fact of the goal is compared with the roots of all subtrees of the (empty) root node of the tree, and if a match is found the element of the goal is skipped and the rest of the goal recursively inserted into the appropriate subtree. If no match is found, a new subtree is created and the remaining facts inserted in a straight path down to the leaf.

Compared with the approach of storing all unsatisfied goals in an array and accessing them using a hashing function or some other type of tree, the goal tree conserves space to a certain extent. The worst-case time complexity of goal retrieval in the current implementation is $O(lf)$, where l is goal length, and f is the number of distinct facts instantiated by the system. Although $f \gg l$ for typical planning problems, the chosen data structure functions efficiently in the plan creation setting since it was empirically observed that the tree usually remains sparse, with the number of children of a node never really nearing f. Attempts to optimize retrieval by performing binary search on children, thus turning the complexity estimate into $O(l \log f)$, resulted in no practical speedup. On average, only several percent of time taken for plan generation is expended on interaction with the goal tree.

On the other hand, the space requirements of the goal tree may become prohibitively high for large planning problems. This calls for further investigation of into methods for (sub)goal compression, as well as search space pruning.

In contrast to many existing planners, APP does not perform *grounding* (creating instances of facts and actions) prior to the execution of the planning algorithm—instances of facts and actions are created when they become needed. The reason behind this approach is that a planning instance represents an agent's model of its environment and may therefore contain a large number of objects. A consequence of possibly having a large number of objects is the expenditure of great amounts of memory, as well as a lot of processing time needed to perform grounding. Creating instances during plan search lessens the memory and time requirements for this phase, but increases the time needed for the creation of deeper levels of the graph.

[2]The currently used heuristic is the number of no-op actions, i.e. the number of matching facts in the prefixes of the goal and subgoal.

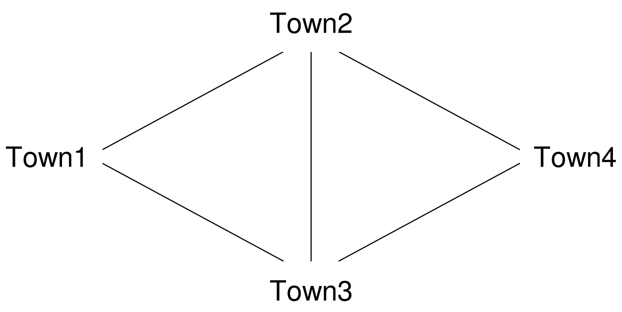


FIG. 4.1. *Towns and roads connecting them*

4. A Working Example. This section introduces a working example that will be used to illustrate APP's properties. The planning domain and planning problem of a simple logistic setting will be described using PDDL2.1 notation.

Let there be four towns (`Town1`, `Town2`, `Town3` and `Town4`) which are connected by roads as in Fig. 4.1. Let there be two trucks (`Truck1` and `Truck2`) and two packages (`Package1` and `Package2`) in `Town1`. The goal for this planning problem is to deliver `Package1` and `Package2` to `Town4` using the given trucks.

In order to describe the *planning domain* for this example, predicates and actions will be defined as follows.

Four predicates will be used:

- `(at ?pkg - package ?town - town)`: is package `?pkg` in town `?town`
- `(in-town ?truck - truck ?town - town)`: is truck `?truck` in town `?town`
- `(in ?pkg - package ?truck - truck)`: is package `?pkg` in truck `?truck`
- `(road ?town1 - town ?town2 - town)`: are towns `?town1` and `?town2` connected by a road

For this planning domain, three actions are needed. Action `load` is used to load package `?pkg` into truck `?truck`. The necessary precondition for this action is that both `?pkg` and `?truck` are in the same town. After the execution of this action, the package will be loaded into the truck.

```
(:action load
   :parameters   (?pkg - package ?truck - truck ?town - town)
   :precondition (and (in-town ?truck ?town) (at ?pkg ?town))
   :effect       (and (not (at ?pkg ?town)) (in ?pkg ?truck)))
```

Action `unload` is used to unload package `?pkg` from truck `?truck`. The necessary precondition for this action is that `?pkg` is in `?truck`, and that `?truck` is in town `?town`.

```
(:action unload
   :parameters   (?pkg - package ?truck - truck ?town - town)
   :precondition (and (in-town ?truck ?town) (in ?pkg ?truck))
   :effect       (and (not (in ?pkg ?truck)) (at ?pkg ?town)))
```

Transportation of packages can be modeled using action `drive` whose preconditions are that `?truck` is in town `?from` and that towns `?from` and `?to` are connected by a road. The effect of action `drive` is that `?truck` is moved to town `?to` (along with all the packages inside the truck).

```
(:action drive
   :parameters   (?truck - truck ?from - town ?to - town)
   :precondition (and (in-town ?truck ?from) (road ?from ?to ))
   :effect       (and (not (in-town ?truck ?from))
                      (in-town ?truck ?to))))
```

The initial situation of a system (that corresponds to an agent's current state) and goals are defined in the *planning problem*. The initial situation in this example is that packages `Package1` and `Package2` and trucks `Truck1` and `Truck2` are in town `Town1`. Also, roads between towns must be defined according to Fig. 4.1. This situation can be expressed using following PDDL code:

```
(:init (at Package1 Town1) (at Package2 Town2)
   (in-town Truck1 Town1) (in-town Truck2 Town1)
   (road Town1 Town2) (road Town2 Town1) (road Town1 Town3)
   (road Town3 Town1) (road Town2 Town3) (road Town3 Town2)
```

```
    (road Town2 Town4) (road Town4 Town2) (road Town3 Town4)
    (road Town4 Town3))
```

The goal in this example is to reach the state where both packages are in Town4. This goal can be defined by:

```
(:goal (and (at Package1 Town4) (at Package2 Town4)))
```

5. The Planning Cycle. APP enables an agent to impose full control over the planning domain and problem. Therefore, APP is divided into four modules as shown in Fig. 3.1(b). Those modules encapsulate data manipulation, plan search, plan execution and manipulation of already created plans.

5.1. Data Manipulation. Data input into a planning instance may be done in two ways: by loading the planning domain and problem from PDDL files, or "manually" from Java code by creating instances of appropriate classes and inserting them into the planning instance. The internal structure of the planning domain and problem in APP are modeled according to the PDDL 2.1 standard [9].

Input of the domain and problem from files is done via methods loadDomain(filepath, filename) and loadProblem(filepath, filename). This way it is possible to load the entire domain and/or problem, and later modify them from Java if necessary. Since the domain is independent of the problem(s), domain input is possible not only on the planning instance level, but also for the whole instance of the planner.

During its life cycle, an agent may obtain information about changes in its environment. As it is of crucial importance for the planning instance to be up to date with the state of the agent's surroundings, the agent is given the capability to insert, change or remove data from a planning instance. This is achieved by making available over 100 classes, the structure of which closely mimics the BNF structure of the domain and planning problem described in [9]. Some of these classes represent basic planning concepts (action, predicate, function, constant, object, etc.), while the majority of classes representing some form of expression used for defining preconditions and effects of actions, as well as defining the goal.

In the working example, during its life cycle an agent can obtain information that the road between Town2 and Town4 has subsequently been closed. After the agent deletes relation (road Town2 Town4) from the planning problem, the planning process needs to be triggered again because the previously generated plans that use this relation are no longer valid. Also, the insertion of this relation back into the planning problem is possible after the road is reopened.

5.2. The Plan Creation Module. There are four basic types of plans that can be used in APP. Together with those types of plans, a set of plans called multiPlan can be used. All plan types and multiPlan can be created by some of the four methods for plan generation.

5.2.1. Basic Types of APP Plans. APP can generate various basic types of plans:

- SingleAction is a plan containing only one instance of an action;
- SerialPlan contains a series of subplans, where the execution sequence of the plans is strictly defined;
- ParallelPlan consists of several subplans, where the execution sequence of the plans is not defined, and subplans may be executed in parallel;
- AlternativePlan is a plan represented as a set of plans. If any subplan is valid, the whole plan is;
- MultiPlan represents a set of plans, where each plan is valid in terms of successfully reaching the goal. MultiPlan was introduced to enable an agent to choose from a set of plans based on its own additional criteria.

Basic plans in APP may be combined in order to create a plan that satisfies the goal defined by the planning problem. Usually, the generated plan is a SerialPlan that contains ParallelPlans as its subplans. Every parallel plan contains more than one action and every action is represented using the SingleAction type. If a plan contains alternative plans, the structure of the plan can become more complex.

5.2.2. Methods for Plan Creation. Planning instances are provided with methods that initiate the creation of an appropriate type of plan, optionally limited by a specified amount of time allowed for the plan generation process.

There are four methods that can be invoked to create a plan in APP:

- createPlan() returns the first plan that is found,
- createMultiPlan() returns a MultiPlan, where all plans are of the same length,
- createAllPlan() returns a MultiPlan with plans of different lengths,
- createAlterPlan() returns a plan that contains alternative plans.

Methods createPlan() and createAlterPlan() are used to create one plan, and the generated plan can be easily executed by calling method execute() of that plan. In contrast to them, methods createMultiPlan() and createAllPlans() are used to create multiple plans, and an agent needs to chose one of the plans that will be executed.

In the logistic example, if an agent calls method createPlan(), the first generated plan will be returned. This plan is given in Fig. 5.1, where steps 1 and 4 represent subplans whose actions can be executed in parallel. Steps 2 and 3 can be represented using singleAction plans, because they contain only one action.

```
1: (load Package1 Truck1 Town1) (load Package2 Truck1 Town1)
2: (drive Truck1 Town1 Town2)
3: (drive truck1 Town2 Town4)
4: (unload Package1 Truck1 Town4) (unload Package2 Truck1 Town4)
```

FIG. 5.1. *The plan generated using method createPlan()*

Within the logistic example, method createMultiPlan() returns a MultiPlan, which consists of serial plans of the same length (four steps). One of the serial plans is depicted in Fig. 5.1. When an agent creates multiple plans, it can later choose which plan is going to be executed. The criteria for than can be how many trucks are used, which trucks are used, the number of action instances, etc.

By using createMultiPlan() it is possible to obtain plan analogous to the plan in Fig. 5.1, where Truck2 is used instead of Truck1. Method createMultiPlan() can also create plans where both trucks are used and the plan given in Fig. 5.2 is one of them.

```
1: (load Package1 Truck1 Town1) (load Package2 Truck2 Town1)
2: (drive Truck1 Town1 Town2) (drive Truck2 Town1 Town3)
3: (drive Truck1 Town2 Town4) (drive Truck2 Town3 Town4)
4: (unload Package1 Truck1 Town4) (unload Package2 Truck2 Town4)
```

FIG. 5.2. *A plan that uses both trucks*

Method createAllPlans() is similar to method createMultiPlan(), with the only difference that the number of steps in generated plans can vary from one plan to another. By using this method, it is possible to obtain a plan that uses towns Town2 and Town3, which can be very useful in situations where, in reality, roads between towns Town1 and Town3, and towns Town2 and Town4 are closed often. One of the plans that uses the road between Town2 and Town3 is given in Fig. 5.3, and in contrast to previous plans it has five steps.

```
1: (load Package1 Truck1 Town1) (load Package2 Truck1 Town1)
2: (drive Truck1 Town1 Town2)
3: (drive Truck1 Town2 Town3)
4: (drive Truck1 Town3 Town4)
5: (unload Package1 Truck1 Town4) (unload Package2 Truck1 Town4)
```

FIG. 5.3. *A plan that uses the road between Town2 and Town3*

Method createAlterPlan() creates a plan that contains alternative plans. By using this method it is possible to obtain a plan that represents an integrated version of plans in Fig. 5.1 and Fig. 5.3. The first two steps of this integrated plan are the same as in the original plans, while the remaining steps form an AlternativePlan that contains two alternatives. These alternatives represent the remaining parts of the original plans, where one of them leads directly to Town4, while the other first leads to Town3 and then to Town4. This integrated plan is given in Fig. 5.4.

5.3. Plan Execution. From the point of view of practical application, one of the main weaknesses of standard planning systems is that after generating the plan they consider the job done. These plans are for the most part not ever executed in a real, possibly changing environment. This was one of the main motivations behind the design of APP.

Every APP plan possesses the method execute() which initiates plan execution. The method returns a boolean value indicating the success or failure of reaching the goal according to the plan. In a static setting the method always returns true for a properly generated plan. But in dynamic environments, changes during the process of plan creation, execution, or in between, may render the plan no longer valid.

SerialPlans are executed starting with the first plan in the sequence, continuing with the next plan after the execution of its predecessor successfully terminates. If the any subplan fails, execution of the whole plan is aborted. With ParallelPlans all subplans are executed simultaneously in their own threads, and the plan is successful if all subplans terminate with success.

```
1: (load Package1 Truck1 Town1) (load Package2 Truck1 Town1)
2: (drive Truck1 Town1 Town2)
{ 3: (drive Truck1 Town2 Town4)
  4: (unload Package1 Truck1 Town4) (unload Package2 Truck1 Town4)
| 3: (drive Truck1 Town2 Town3)
  4: (drive Truck1 Town3 Town4)
  5: (unload Package1 Truck1 Town4) (unload Package2 Truck1 Town4)}
```

FIG. 5.4. *The plan created using createAlterPlans()*

During execution of AlternativePlans, if a subplan is not executed successfully, the next alternative is tried. An AlternativePlan is successfully executed if at least one of its subplans is successful. This type of plan may be used within an integrated version of a MultiPlan.

When the plan given in Fig. 5.1 is executed, both actions for loading packages in step 1 will be executed in parallel. After that, the actions from steps 2 and 3 will be executed sequentially. Finally, the actions from step 4 will be executed in parallel.

During the execution of an alternative plan, if it is not possible to execute the first alternative, APP will try to execute the following one. In the plan given in Fig. 5.4, during the execution of action `(drive Truck1 Town1 Town2)`, the road between `Town2` and `Town4` can unexpectedly be closed. Since the first alternative uses relation `(road Town2 Town4)`, its execution will fail and the second alternative will be tried.

5.3.1. Execution of a SingleAction Plan. Every APP plan is, at its core, composed of atomic actions represented by a SingleAction plan. To enable the execution of an action in the actual environment of an agent, every action from a planning domain is assigned an instance of class ActionCode, which includes the method run(action, agent). This method is invoked at the execution of the SingleAction plan.

ActionCode is an abstract class whose method run(action, agent) should be overridden by a method containing code to be executed in an agent's environment. After inheriting the class and creating an instance, it is necessary to assign this instance to an appropriate action within the domain. This is done via method insertActionCode(name, actionCodeInstance), where name denotes the name of the action. Within the run method an agent may execute arbitrary Java code and do whatever is in line with performing the action in his practical environment. Parameter action may be used to access various properties of an action, e.g. its name and arguments. Parameter agent allows access to the resources of an agent.

APP automatically insures that the planning instance is consistent with the state of the environment. After every executed action, APP updates the planning problem according to the effects of the action, enabling the addition of new, as well as change and deletion of existing data.

In the logistic example, three classes that inherit class ActionCode should be created, one for each action in the domain. Method run() in the class that corresponds to action `load` will contain statements for turning on the crane and statements for loading packages using that crane. With the class that corresponds to action `unload` the situation is similar. Inside the body of method run() of the class that corresponds to action `drive` an agent can, for example, check the level of fuel in a truck or display truck movement to the user.

5.4. Properties of APP Plans. Besides plan execution, APP offers some other plan-related functionality. On a standard level, it is possible to check the number of actions in a plan, copy plans, check if two plans are equal and convert a plan into text for easy output. In addition, it is possible to convert plans from one form into another. For example, method mergePlans(multiPlan) transforms the set of independent subplans within the multiPlan into plans which may contain AlternativePlans.

APP allows access to every part of a plan, that way enabling enquiry into certain properties of plans which can not be specified as constraints in the planning domain. The creation of MultiPlans facilitates this functionality by enabling an agent to generate a large number of plans and then filter them according to additional criteria. Such criteria may include the least use of some action which is too costly for the system, avoiding the use of a particular error prone resource, discouragement of executing plans similar to other plans which had failed in the past, etc.

Let us assume that the road between `Town1` and `Town3` in the working example can very often become impassable. By using method createAllPlans(), an agent can obtain a set of plans and exclude all plans which use that road. Subsequently, the agent can choose one of the plans that are not excluded and execute it. If the execution of the plan fails, the agent can exclude all invalid plans and choose one of the remaining valid ones. This strategy can be less time consuming than restarting the search process. A simplified version of this strategy is to call method mergePlans() which will integrate

```
. . .
Planner logistic = new Planner(thisAgent);
logistic.loadDomain("", "domain.pddl");
int instanceNum = logistic.addInstance();
logistic.getInstance(instanceNum).loadProblem("", "problem.pddl");
logistic.getDomain().insertActionCode("load", new load());
logistic.getDomain().insertActionCode("unLoad", new unLoad());
logistic.getDomain().insertActionCode("drive", new drive());
Plan p = logistic.getInstance(instanceNum).createPlan();
ok = p.execute();
```

FIG. 6.1. *Code of a logistic agent that uses APP*

all plans from the generated set and create one plan that contains alternatives, thus automating the process of subplan exclusion during execution.

6. Implementation of a Logistic Agent. One of the main design ideas behind APP was to create a planning system that will be easy to use and provide a large number of services to agents.

The code shown in Fig. 6.1 presents one of the simplest ways to use APP inside an agent's body. An agent should first create one instance of the planner, and load a planning domain. After that, the agent should create a planning instance, load a planning problem, and connect actions `load`, `unload` and `drive` with code to be executed during the execution of those actions. Finally, the agent should initiate plan search and execute the generated plan. If an action is not explicitly connected with code, it will be connected with empty code, and the only effect of action execution will be the update of the planning problem which represents the agent's knowledge about its environment.

This implementation of a logistic agent is overly simple and can cause problems during plan execution in dynamic environments. If one of the used roads is closed during the execution of some previous actions, execution of this plan will fail and variable ok will be set to false. There are three ways in APP to prevent this kind of failure.

The first way is trivial and requires a new plan to be generated and executed. This strategy can be pretty bad because the planning process can be very time costly. The implementation of this strategy requires replacing the last two rows of the previous implementation with following code:

```
ok = false;
while (!ok) {
    Plan p = logistic.getInstance(instanceNum).createPlan();
    ok = p.execute();
}
```

The second and third ways utilize generation of multiple and alternative plans. If an agent generates multiple plans, it can decide which plan is going to be executed. If the execution of a plan fails, the agent can choose some other plan from the generated set of plans. The problem with this strategy is that many of those plans will cease to be executable, because several of their actions were already executed as parts of plans that previously failed, and such actions can not be executed again. In this case it is possible to truncate those actions, and to execute the rest of the plan. This strategy requires replacing the last two rows from the code in Fig. 6.1 with following code:

```
MultiPlan mp = logistic.getInstance(instanceNum).createMultiPlan();
ok = false;
while (!ok && mp.numberOfPlans()!=0) {
    Plan p = mp.getNthSubPlan(1);
    mp.removeNthSubPlan(1);
    Plan tmp = Adjust(p);
    ok = tmp.execute();
}
```

Plans with alternatives are one of APP's notable features. During their execution there is no need check whether the first several actions were already executed, because this will be done automatically by the module for plan execution and monitoring. If an agent wants to use alternative plans it should invoke method createAlterPlan() in its original implementation instead of method createPlan(). The last two rows in this version of the agent are as follows:

```
Plan p = logistic.getInstance(instanceNum).createAlterPlan();
ok = p.execute();
```

6.1. Agent Implementation using Two Instances of a Planner. One of the advantages of APP is the possibility of using more than one planning domain at the same time. These domains are completely independent and every domain is defined inside a new instance of APP's planner, as shown in Fig. 3.1 (Agent C).

If a logistic agent can use a crane to (un)load packages, it can use a planning system to plan actions for (un)loading, as mentioned in Section 5.3. If an agent can not use more than one domain, the only way to utilize a classical planning system for two domains is to create an integrated version of domains. This way the number of actions, predicates, objects and relations grows and can rapidly increase the amount of used memory and time in the plan generation phase. Because of this, there is a large possibility that the system will fail to find a plan, especially if the amount of allotted time is small.

By using APP, it is possible to split the planning domain of the logistic example into two domains by moving all actions and data used for (un)loading packages to a new domain inside a new instance of the planner. Since the amount of data in every domain is roughly one half of the integrated domain, the time and memory used to generate a plan can rapidly decrease because the dependence between time and the amount of data is stronger than linear.

As was explained in Section 5.3, plan generation for package loading can be done during the execution of action `load`. Inside the body of method run() for action `load`, an agent can call method createPlan() for the crane domain, and execute the generated plan. It is possible to create a new instance of a planner, or to use the one that is defined inside the agent.

Figure 6.2 shows class load that implements the loading of a package into a truck. Variable crane refers to the instance of the planner for the crane domain which is created inside the agent's body. Subsequently, method createPlan() is called to create a plan for package loading, and method execute() is called to execute the plan.

```
class load extends actionCode {
    public boolean run(actionInstance action, Object agent) {
        Planner crane = ((Agent) agent).crane;
        Plan p = crane.getInstance(instanceNum).createPlan();
        return p.execute();
    }
}
```

FIG. 6.2. *Class load that loads a package into a truck using a crane*

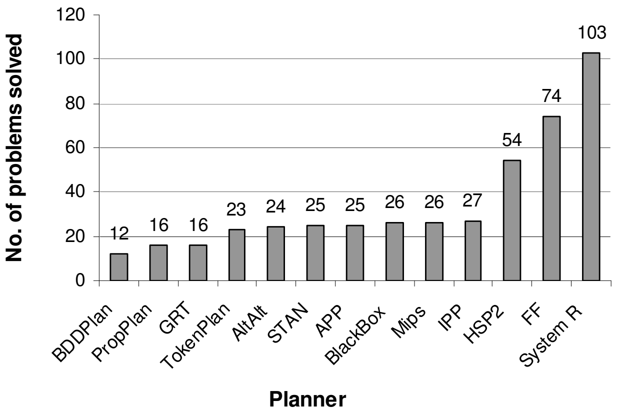
7. Experimental Results. Although the initial objectives behind APP did not include competing with the state-of-the-art planners, APP has proven to be more efficient than anticipated. In order to evaluate its efficiency, APP was tested on examples from the planning competition of the Artificial Intelligence Planning Systems Conference (AIPS) in 2000. This particular benchmark was chosen because APP supports a subset of PDDL 2.1, which was the official language of the AIPS 2000 competition.

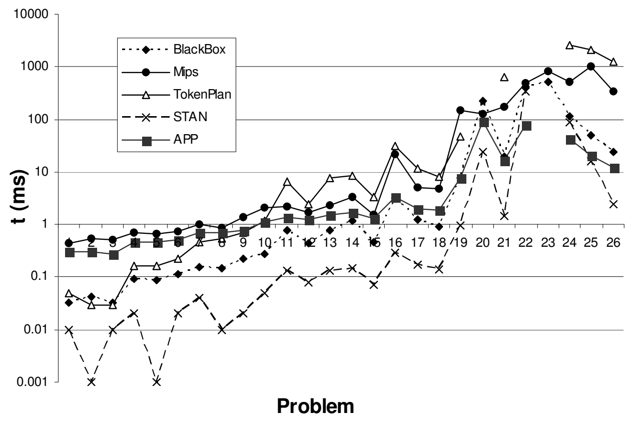
Figure 7.1 shows the results of testing in the *blocks* domain. Figure 7.1(a) depicts the number of solved problems for various planners, while Fig. 7.1(b) shows the time in milliseconds taken by each planner to solve a particular problem. The results for other planners were obtained from the official Web site of the conference [1], while APP was tested on a closely reproduced computer configuration, with the same time limit of 30 minutes for each problem.

Figure 7.1(a) shows that APP is a medium-power planner judging by the number of solved problems. From the first 26 problems APP could not solve only "blocks-11-1" which was problematic for all planners except the three most efficient. IPP solved two problems more than APP, but the time expended on the two was an order of magnitude longer than the time IPP typically took to solve the other 25 problems.

Figure 7.1(b) focuses on the time spent by the medium-power planners on the first 26 problems. The fastest planner was STAN which is well known for its efficient implementation, although APP did show a better result on "blocks-11-0" and "blocks-11-2" problems. On simpler problems, APP exhibited about the same efficiency as Mips, while on more difficult ones APP proved somewhat superior. TokenPlan and BlackBox achieved better results on simpler problems, but as problems grew more difficult APP was more up to the task.

Such behavior of APP can be explained by the fact that for smaller problems more time is spent on the creation of the planning graph, while larger problems take up more time on graph searching. As APP creates instances of actions during graph creation, this could be expected. Planning systems which first create all instances of actions, and than search for a





(a) (b)

FIG. 7.1. *Experimental results showing (a) the number of successfully solved problems for every planner, and (b) the time that medium-power planners spent on each of the first 26 problems*

plan, achieve better results in these situations. However, real world systems, for which APP is designed, usually contain a large number of objects and actions, punishing this approach to action instantiation. The reason behind APP's better results for larger problems in the benchmark lies a great deal in the use of the goal tree data structure. The goal tree speeds up the check for prior attempts to satisfy a goal, which is important for larger graphs.

It is important to note that unlike most other planners, APP does not use heuristics to prune unpromising paths during the search of the graph. A good consequence of this is that APP is guaranteed to find the shortest possible plan for any given problem, which puts APP among the most efficient systems judging by this criterion. The bad consequence is that search time is greatly increased.

8. Conclusions and Future Work. This paper presented the APP Java package which enables agents to expand their capabilities with automatic planning. We have shown how APP can be integrated into a multi-agent system, and how its communication with an agent may be achieved.

The basic services APP offers to an agent include creating multiple planning instances within one instance of a planner, and accessing the data within a planning instance in order for existing data to be read, changed or deleted, and new data to be inserted. In contrast to APP, most of the other planners allow only one planning instance, and are not able to modify it after the plan had been created. Furthermore, because of its ease of data manipulation, APP may be utilized to implement agents' representations of the current state of the world.

Besides plan creation, explicit plan *execution* is also supported, facilitating the use of APP in dynamic environments. APP also provides services for checking various properties of plans, which is a feature not supported by the majority of planners.

APP uses a modified algorithm for plan creation which employs a planning graph and a goal tree structure, supporting the generation of many different types of plans. For example, a combination of SerialPlans and ParallelPlans may be used to obtain partial order plans (POP), which are supported by the majority of planners, while AlternativePlans allow execution of multiple subplans until one terminates with success. AlternativePlans may be combined with SerialPlans and ParallelPlans to derive plans which have increased chances for behaving robustly in dynamic environments.

Currently, APP supports a subset of PDDL 2.1 which includes :strips, :typing and :fluents requirements. Upgrading to the full :adl set is underway, as well as extension to the third and fourth level of PDDL which incorporate actions with duration. In order to provide faster plan generation, research into heuristics will be done to enable reliable and efficient pruning of unpromising paths during graph search.

Since the main purpose of APP is to serve as a library for creation of intelligent agents, we intend to add a number of new methods for investigating various properties of plans. One example is a method which simulates plan execution without actually firing any actions in the agent environment, in order to check the validity of a plan before actually executing it.

REFERENCES

[1] *Aips-00 planning competition.* `http://www.cs.toronto.edu/aips2000/`.

[2] M. Badjonski, M. Ivanović, and Z. Budimac, *Adaptable Java Agents (AJA): A tool for programming of multi-agent systems*, SIGPLAN Notices, 40 (2005), pp. 17–26.

[3] F. Bellifemine, G. Caire, and D. Greenwood, *Developing Multi-Agent Systems with JADE*, John Wiley & Sons, 2007.

[4] A. L. Blum and M. L. Furst, *Fast planning through planning graph analysis*, Artificial Intelligence, 90 (1997), pp. 281–300.

[5] B. Bonet and H. Geffner, *Heuristic search planner 2.0*, AI Magazine, 22 (2001), pp. 77–80.

[6] M. Do and S. Kambhampati, *SAPA: A multi-objective metric temporal planner*, Journal of Artificial Intelligence Research, 20 (2003), pp. 155–194.

[7] S. Edelkamp and J. Hoffmann, *PDDL2.2: The language for the classical part of the 4th international planning competition*, Tech. Report 195, Institute of Computer Science, University of Freiburg, 2004.

[8] R. Fikes and N. Nilsson, *STRIPS: A new approach to the application of theorem proving to problem solving*, Artificial Intelligence, 2 (1971), pp. 189–208.

[9] M. Fox and D. Long, *PDDL2.1: An extension to PDDL for expressing temporal planning domains*, Journal of Artificial Intelligence Research, 20 (2003), pp. 61–124.

[10] A. Gerevini and D. Long, *Plan constraints and preferences in PDDL3*, tech. report, Department of Electronics for Automation, University of Brescia, 2005.

[11] A. Helsinger and T. Wright, *Cougaar: A robust configurable multi agent platform*, in Proc. IEEE Aerospace Conference, 2005, pp. 1–10.

[12] J. Hoffmann and B. Nebel, *The FF planning system: Fast plan generation through heuristic search*, Journal of Artificial Intelligence Research, 14 (2001), pp. 253–302.

[13] S. Kambhampati, E. Parker, and E. Lambrecht, *Understanding and extending Graphplan*, in Proc. ECP'97, 4th European Conference on Planning, LNCS 1348, 1997, pp. 260–272.

[14] D. Long and M. Fox, *Efficient implementation of the plan graph in STAN*, Journal of Artificial Intelligence Research, 10 (1999), pp. 87–115.

[15] M. Ghallab et al., *PDDL – the planning domain definition language*, Tech. Report CVC TR-98-003, Yale Center for Computational Vision and Control, 1998.

[16] D. S. Nau, T. C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman, *SHOP2: An HTN planning system*, Journal of Artificial Intelligence Research, 20 (2003), pp. 379–404.

[17] D. S. Nau, Y. Cao, A. Lotem, and H. Muñoz Avila, *SHOP: Simple hierarchical ordered planner*, in Proc. IJCAI'99, 16th International Joint Conference on Artificial Intelligence, 1999, pp. 968–973.

[18] A. Newell, J. C. Shaw, and H. A. Simon, *Report on a general problem-solving program*, in Proc. International Conference on Information Processing, 1959, pp. 256–264.

[19] E. P. D. Pednault, *ADL: Exploring the middle ground between STRIPS and the situation calculus*, in Proc. KR'89, First International Conference on Principles of Knowledge Representation and Reasoning, 1989, pp. 324–332.

[20] R. Sanchez-Nigenda, X. Nguyen, and S. Kambhampati, *AltAlt: Combining the advantages of Graphplan and heuristic state search*, in Proc. KBCS'00, 3rd International Conference on Knowledge-based Systems, 2000.

[21] R. Sedgewick, *Algorithms in Java, Parts 1–4*, Addison Wesley, 3rd ed., 2002.

Edited by: Maria Ganzha, Marcin Paprzycki
Received: Jan 28, 2008
Accepted: Feb 9, 2008

SCALABLE COMPUTING: PRACTICE AND EXPERIENCE
Volume 9, Number 1, pp. 51–59. http://www.scpe.org
ISSN 1895-1767
© 2008 SCPE

KNOWLEDGE PROCESSING FOR WEB SEARCH—AN INTEGRATED MODEL AND EXPERIMENTS

P. GURSKÝ* AND T. HORVÁTH*, J. JIRÁSEK*, R. NOVOTNÝ*, J. PRIBOLOVÁ*, V. VANEKOVÁ* AND P. VOJTÁŠ†

Abstract. We propose a model of a middleware system enabling personalized web search for users with different preferences. We integrate both inductive and deductive tasks to find user preferences and consequently best objects. The model is based on modeling preferences by fuzzy sets and fuzzy logic. We present the model-theoretic semantics for fuzzy description logic f-$\mathcal{EL}$ which is the motivation of creating a model for fuzzy RDF. Our model was experimentally implemented and the integration was tested.

Key words. middleware, fuzzy DL, fuzzy RDF, relevant objects, user preferences

1. Introduction and Motivation. One of the main goals of semantic web is to enable easy and automatic access to web resources and services by middleware engines or agents. Our research leads to the model of a middleware system which will help users in searching for objects from heterogenous sources but a single domain. In this paper we present different approaches to the most important aspect of such system—retrieving the best objects according to user's preferences.

Let us consider the following example: imagine a user looking for a hotel which has *good price*, *good distance from an airport* and has *good equipment in rooms*.

Each user (or group of users) has his own sense of quality (i. e. the preference). For the price of hotels, one user could prefer cheap hotels (student), second prefers expensive hotels (manager) and the other one prefers middle price (professor). The underlying meaning of this ordering is that the user determines the relations "better" or "worse" between two values of a given property. For each such property, user has a notion about the ordering of objects based on real value of property from an attribute domain. We call these orderings of particular attribute domains the *user local preferences*.

Nevertheless, these local preferences usually lead to incomparable objects, e. g. one hotel that is cheaper than another (which means that the latter is better for a student), but has inferior room equipment. The combination of local preferences gives global preference and will be modeled by fuzzy aggregation operators.

The main contributions of this paper are integration of several methods for local and global preferences into one framework, and practical introductory experiment with our system.

The paper is organized as follows: section 2 introduces methods for detecting user preferences and searching for relevant objects and it provides a theoretical model of these preferences based on description logic. Section 3 describes the system and experiments. Finally, section 4 concludes and provides some plans for future research.

2. Models and Methods. In this chapter we describe models and methods we use for a solution of the problem of web search. Later these methods are implemented and tested.

2.1. Detecting Local Preferences. To learn local preferences, we have several possibilities. First is, we present to user a representative sample of hotels (see Table 2.1) with several attributes, e. g. distance from the airport, price of the accommodation and equipment of rooms. The user classifies hotels into categories *poor*, *good* and *excellent* according to the relevance of the hotel to him. In practice we have chosen the seven classes of Likert scale.

We distinguish four basic types of local preferences, according to user's most preferable attribute values. We call these basic types *higher-best*, *lower-best*, *middle-best* and *marginal-best.*

The local preferences can be detected by statistical methods, e. g. regression or QUIN [4]. Using the linear regression we can detect just two basic types (*higher-best* and *lower-best*). In contrast to QUIN, the regression is resistant against statistically irrelevant values. Thus for the purposes of detecting the aforementioned four basic types, the polynomial regression is the most appropriate approach. In case of the Table 2.1, we checked that the higher distance (*higher-best* type of ordering) and the lower price (*lower-best* type of ordering) are appropriate for the user. Albeit this method is not yet used in the experimental implementation, it is useful for the motivation purposes.

The second possibility is to learn local preferences explicitly from the user. The user has possibility to specify his preferences by explicit choice of fuzzy functions. We have used this method in our experiments (see Section 3).

2.2. Learning Global Preferences. We learn user's global preferences by the method of ordinal classification with monotonicity constraints [9] based on Inductive Logic Programming (ILP) system ALEPH [15].

*Institute of Computer Science, Šafárik University, Košice, Slovakia, ({name.surname}@upjs.sk).
†Department of Software Engineering, Charles University, Prague, Czech Republic, (Peter.Vojtas@mff.cuni.cz).

TABLE 2.1
Representative sample of hotels evaluated by the user

Hotel	Distance	Price	Equipment	Evaluation
Apple	100 m	$ 99	nothing	poor
Danube	1300 m	$ 120	TV	good
Cherry	500 m	$ 99	internet	good
Iris	1100 m	$ 35	internet, TV	excellent
Lemon	500 m	$ 149	nothing	poor
Linden	1200 m	$ 60	internet, TV	excellent
Oak	500 m	$149	internet, TV	good
Pear	500 m	$ 99	TV	good
Poplar	100 m	$ 99	internet, TV	good
Rhine	500 m	$ 99	nothing	poor
Rose	500 m	$ 99	internet, TV	excellent
Spruce	300 m	$ 40	Internet	good
Themse	100 m	$ 149	internet, TV	poor
Tulip	800 m	$ 45	internet, TV	excellent

The user's global preferences are computed by using his local preferences which can be well represented in ILP. For illustration, consider that we have discretized values of distance to three classes: *near*, *middle* and *far*. The different orderings of these classes for different users can be:

$$\begin{array}{rcccl} \text{near} & \leq & \text{middle} & \leq & \text{far} \\ \text{near} & \leq & \text{far} & \leq & \text{middle} \\ \text{far} & \leq & \text{middle} & \leq & \text{near} \\ \text{far} & \leq & \text{near} & \leq & \text{middle} \\ \text{middle} & \leq & \text{near} & \leq & \text{far} \\ \text{middle} & \leq & \text{far} & \leq & \text{near} \end{array}$$

An usual aggregation function can be easily simulated by monotone classification rules in the sense of many valued logic (on Figure 2.1 the computed user's global preferences from our illustrative data are presented):

$$\begin{array}{lcl} \text{evaluation} & = & \text{excellent IF distance } \geq 500 \text{ AND price } \leq 99 \\ & & \text{AND services } = \{\text{TV, internet}\} \\ \text{evaluation} & = & \text{good IF (distance } \geq 500 \text{ AND services} = \{\text{TV}\}) \\ & & \text{OR (price } \leq 99 \text{ AND services} = \{\text{internet}\}) \end{array}$$

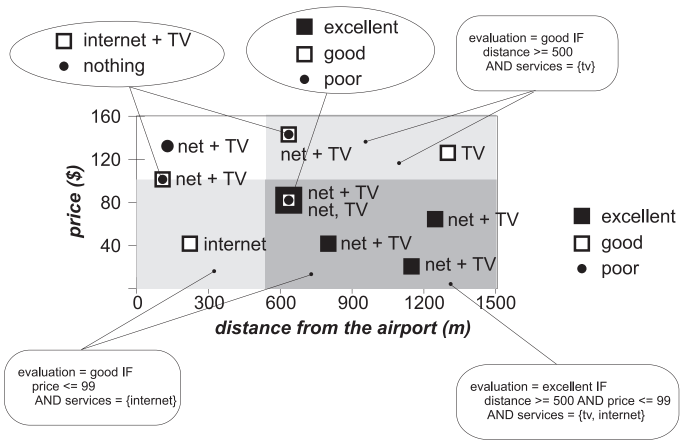


FIG. 2.1. *The results of our approach in the case of our illustrative example*

We can see that the ordered meaning of the classification is preserved in our results (this is proved in [10] as the *igap-consistency* of our approach): hotels classified in the grade "excellent" by the user also fulfills requirements for "good" and "poor" hotels, and "good" hotels fulfills requirements for "poor" ones (e. g. the hotel with 800 m distance from airport and \$45 price equipped with internet and TV is "at least as appropriate as" the hotel 300 m far from the airport and \$40 price equipped just with internet) according to the local preferences (more far and more cheap is the better).

From our results we can obtain also additional information about crisp attributes (e. g. equipment). The meaning of any classification rule is as follows: If the attributes of object x fulfill expressions on the right side of the rule (body) then the overall value of x is at least the same as on the left side of the rule (head). We can, of course, assign the explicit values to vague concepts like excellent or good. During the simulation of computation of aggregation function we can simply test the validity of requirements of the rules from the strongest rule to weaker ones. When we find the rule that holds, we can say that the overall value of the object is the value on the left side of the rule. Since we test the sorted rules, we always rank the object with the highest possible value.

2.3. Fuzzy RDF Based on Fuzzy Description Logic. In this section we analyze the model of fuzzy RDF/OWL based on a model of fuzzy description logic. The terms like "cheap", "expensive" or "near" represent fuzzy sets. Our model of fuzzy RDF includes such fuzzy sets stored as RDF triples.

One important feature of our model is that we can prepare fuzzy RDF independently from user global preference. This is because we can adapt to user by adjusting his aggregation function $@$. This makes the processing of data more effective, because we do not need to order data for every user query. We already have the data ordered and we just combine the relevancies from fuzzy RDF into one result.

We introduce the model of building fuzzy RDF based on fuzzy description logic f-$\mathcal{EL}$ proposed in [17]. This logic removes some features of both classical and fuzzy description logic (like negation, universal restriction and fuzzy roles). On the other hand it adds an aggregation operator $@$. It should be also noted, that we lose the ability to describe fuzziness in roles. However, our data from the domain ontology are crisp (we do not consider uncertainty in values). User preferences are represented as fuzzy concepts and they are the source of fuzziness in results. Thus, we gain combination of particular user preferences to a global score by his $@$ function. An advantage of this description logic is lower complexity of querying. Expressivity is lower than that of full fuzzy description logics (DL) but still sufficient for our task and embedability into web languages and tools (see [16]).

Concepts and roles are the basic building blocks of every description logic. Here, roles express properties of resources, in our case hotels. Although the basic model of expressing RDF triples $\langle$*subject*, *predicate*, object$\rangle$ corresponds to oriented graphs, we use here the language of logic: *predicate*(*subject*, *object*).

The alphabet consists of sets $\mathcal{N}_C$ of concepts names, $\mathcal{N}_R$ role names and $\mathcal{N}_I$ instance names. The roles in f-$\mathcal{EL}$ are crisp and the concepts are fuzzy. Our language of description logic further contains a constructor $\exists$ and a finite set of aggregation functions symbols $@_U$ for each user and/or for each group of users. Concept descriptions in f-$\mathcal{EL}$ are formed according to the following syntax rules

$$C \to \top \mid A \mid @(C_1, \dots C_n) \mid \exists r.C$$

In order to give this syntax a meaning, we have to define interpretations of our language. In f-$\mathcal{EL}$ we have interpretations parameterized by a (possibly partially, usually linearly) ordered set of truth values with aggregations. For a preference structure (a set of truth values $P = [0, 1]$), a P-interpretation is a pair $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \bullet^{\mathcal{I}} \rangle$ with nonempty domain $\Delta^{\mathcal{I}}$ and fuzzy interpretation of language elements:

- $A^{\mathcal{I}} : \Delta^{\mathcal{I}} \to P$, for $A \in \mathcal{N}_C$
- $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, for $r \in \mathcal{N}_R$
- $(\exists r.C)^{\mathcal{I}} = \sup\{C^{\mathcal{I}}(y) : (x, y) \in r^{\mathcal{I}}\}$
- $(@(C_1, \dots, C_n))^{\mathcal{I}}(x) = @^{\bullet}(C_1^{\mathcal{I}}(x), \dots, C_n^{\mathcal{I}}(x))$

Suppose that we have $\mathcal{N}_R = \{\textit{price}, \textit{distanceFromAirport}\}$ where both elements of this set represent the RDF predicates from the domain ontology.

$\mathcal{N}_C = \{\textit{cheap}_U, \textit{close}_U\}$ where $\textit{cheap}_U$ and $\textit{close}_U$ are fuzzy functions explicitly figured by user preference ontology.

$\mathcal{N}_I = \{\textit{Apple}, \textit{Danube}, 99, 120, \dots\}$.

In Herbrand-like interpretation $\mathcal{H}$ we have:

$$\textit{Apple}^{\mathcal{H}} = \text{Apple}, \textit{Danube}^{\mathcal{H}} = \text{Danube}, 99^{\mathcal{H}} = 99, 120^{\mathcal{H}} = 120$$
$$\textit{cheap}_U^{\mathcal{H}}(99) = 0.53, \textit{cheap}_U^{\mathcal{H}}(120) = 0.42$$
$$\textit{price}^{\mathcal{H}} = \{(\text{Apple}, 99), (\text{Danube}, 120)\}$$

For the sake of simplicity we overload our concept *cheap*$_U$ also for hotels (usually we must create new concepts in this case e. g. *cheapHotel*$_U$):

$$\begin{aligned} \mathit{cheap}_U^{\mathcal{H}}(x) &= (\exists \mathit{price}.\mathit{cheap})^{\mathcal{H}}(x) \\ &\text{then} \\ \mathit{cheap}_U^{\mathcal{H}}(\text{Apple}) &= \sup\{\mathit{cheap}_U^{\mathcal{H}}(y) : (\text{Apple}, y) \in \mathit{price}^{\mathcal{H}}\} = 0.53 \\ \mathit{cheap}_U^{\mathcal{H}}(\text{Danube}) &= \sup\{\mathit{cheap}_U^{\mathcal{H}}(y) : (\text{Danube}, y) \in \mathit{price}^{\mathcal{H}}\} = 0.42 \end{aligned}$$

The supremum affects the case when we have more different prices for one hotel. Then we take the highest result.

$$\begin{aligned} (@(\mathit{cheap}_U, \mathit{close}_U))^{\mathcal{H}}(\text{Apple}) &= @^{\bullet}(\mathit{cheap}_U^{\mathcal{H}}(\text{Apple}), \mathit{close}_U^{\mathcal{H}}(\text{Apple})) \\ &= \frac{(3 \times \mathit{close}_U^{\mathcal{H}}(\text{Apple}) + 2 \times \mathit{cheap}_U^{\mathcal{H}}(\text{Apple}))}{5} \end{aligned}$$

Fuzzy description logic f-$\mathcal{EL}$ is the motivation for creating a model for fuzzy RDF. For example, a fuzzy instance $\mathit{cheap}_U^{\mathcal{H}}(\text{Apple}) = 0.53$ can be modelled by the RDF triple: $\langle \text{Apple}, \mathit{cheap}_U^{\mathcal{H}}, 0.53 \rangle$.

This is an embedding of a fuzzy logic construct into classical RDF, which needs to translate also constructions of DL, namely $\exists$*price.cheap* in fuzzy DL turns to composition of roles, where $\exists \mathit{price}.(\exists \mathit{cheap}.\ \top)$ needs an aggregate max (or top-k) extending DL with a concrete domain (see [1]). It is out of the scope of this paper to describe such DL with a concrete domain and embedding of our fuzzy DL in more detail. What we claim here is an experimental implementation of both our fuzzy DL, special crisp DL with a concrete domain and a related model of RDF with extended syntax and semantics of `owl:someValuesFrom`).

Now, in our model, we can specify user profiles and represent these profiles in Fuzzy RDF triples based on data from domain ontology. Our next task is to find relevant objects for individual users.

Consider the type of user preferring cheap hotels. There can be many users that prefer cheap hotels. We want to share the same instance of the class *cheap* for them. However, their notion of "cheapness" can be different. For example one user can say that cheap hotels have price lower than \$30, while for some other user cheap hotel ends at \$50. Fortunately we can easily change the overall evaluation of objects to reflect these individual requirements. In the following theorem $f_1, \ldots, f_n$ are functions that represent local preferences expressed in user preference ontology. Function values $f_1(x_1), \ldots, f_n(x_n)$ are literals from fuzzy RDF expressing the relevance of respective values of an object. We show that we can use the same functions for different users with the same preference ordering and adjust the aggregation function only.

DEFINITION 2.1. *Let D be a subinterval of real line and f be a bijective (either strictly increasing or strictly decreasing) fuzzy function $f = ax + b$, $a \neq 0$, $f : D \to [0, 1]$ and $g : D \to [0, 1]$. We say, that g preserves the ordering of f over D, if for all $x, y \in D$ $f(x) < f(y)$ implies $g(x) \leq g(y)$.*

THEOREM 2.2. *Let $f_1, \ldots, f_n$ be bijective fuzzy functions such that for each $i : f_i = a_i x + b_i$, $a_i \neq 0$, $f_i : D_i \to [0, 1]$. Let $g_1, \ldots, g_n$ be partially linear functions such that they preserve the ordering of $f_1, \ldots, f_n$ over D. Let @ be an n-ary aggregation function. Then there exists n-ary aggregation function $@'$ such that*

$$(\forall x_1, \ldots, x_n) @(g_1(x_1), \ldots, g_n(x_n)) = @'(f_1(x_1), \ldots f_n(x_n)).$$

Proof. The theorem above says about existence of an aggregation function $@'$. We will show how to find this function.

We can assume that f_i and g_i are defined on the same unit interval $D = [0, 1]$. Function g_i is linear over certain subintervals of $[0, 1]$. We take one such subinterval and name it K_j. The following holds for f_i and g_i over K_j:

$$(\forall x_i \in K_j) f_i(x_i) = a_i(x_i) + b_i \text{ and } g_i(x_i) = c_i^j(x_i) + d_i^j$$

Now we define new function h_i for each $i \in [1, n]$ as:

$$(\forall i, j) h_i(y) = c_i^j \frac{(y - b_i)}{a_i} + d_i^j.$$

The aggregation function $@'$ is $@'(y_1, \ldots, y_n) = @(h_1(y_1), \ldots, h_n(y_n))$. If we substitute $f_i(x)$ for every y_i:

$$@'(f_1(x), \ldots, f_n(x)) = @(h_1(f_1(x)), \ldots, h_n(f_n(x_n)))$$

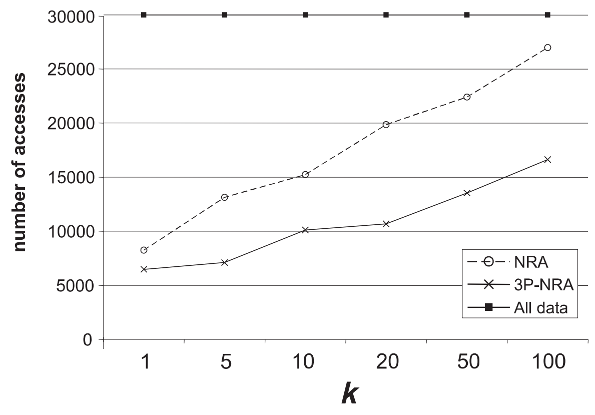


FIG. 2.2. *Number of accesses needed to retrieval of top-k objects*

For arbitrary subinterval $K_j \subseteq [0,1]$ where every g_i is linear we get

$$h_i(f_i(x_i)) = c_i^j \frac{f_i(x_i) - b_i}{a_i} + d_i^j = c_i^j \frac{a_i x_i + b_i - b_i}{a_i} + d_i^j = c_i^j x_i + d_i^j = g_i(x_i)$$

Therefore it holds

$$\begin{aligned} @'(f_1(x_1), \ldots, f_n(x_n)) &= @(h_1(f_1(x_1)), \ldots, h_n(f_n(x_n))) \\ &= @(g_1(x_1), \ldots, g_n(x_n)) \end{aligned}$$

□

This theorem allows users to specify their own meaning of "cheapness" exactly by fuzzy function and similarly for other attributes.

2.4. Relevant Object Search. Now we have orderings of properties from learning of local preferences, rules from learning of global preferences and prepared ordered data stored in fuzzy RDF. Our last task is to find top k objects which are suitable for particular user. We use the extension of middleware search of Ronald Fagin [6]. The main idea is to browse only necessary data until the system is sure that it has top-k objects already. Thus we do not need to calculate with whole data from domain ontology. Limiting the retrieval to the k best objects is often sufficient for the user and it saves time. The time efficiency grows with the number of objects stored in domain ontology and also with the number of properties we consider.

The model in [6] works with data stored in possibly distributed lists that are ordered from the best to the worst in particular property. Fagin considered two kinds of accesses to lists: sorted and random access. The sorted access gets the next best object from the list after each access, so we can retrieve data ordered from the best to the worst. The random access asks for the value of a particular property it includes for a particular object. We want to minimize the number of accesses to lists and of course the time of searching. The random access has one a large drawback. Each random access requires searching of the value of specific object. In the case of sorted access the results are prepared immediately (values can be preordered according to various criteria). Moreover, data can be sent in blocks. These important differences are the reason that we prefer the sorted access algorithms to the random access in our implementation.

In our application we use 3P-NRA (3 Phased No Random Access) algorithm [8], which is an improvement of NRA [7] algorithm.

As we found in our experiments (see Figure 2.2), using the techniques of top-k search can significantly reduce the number of accesses and correspondingly decreases the search time, especially in the case of large set of objects.

In the experiments, the data were generated with various distributions of values. We have used 2 exponential and 2 logarithmic distributions of properties with 10000 objects and 6 types of aggregation functions. For all experiments we have considered 3 properties. Various combinations of properties and aggregation functions were used for composition of 25 inputs for algorithms. The final results show the averages of particular results.

3. Web Search and Experiments. Our models and methods were implemented, integrated and experimentally tested in the project NAZOU, atop the application domain of job offers.

To identify suitable objects for user we need to obtain his local and global preferences. The complete user dependent searching process is illustrated on Figure 3.1.

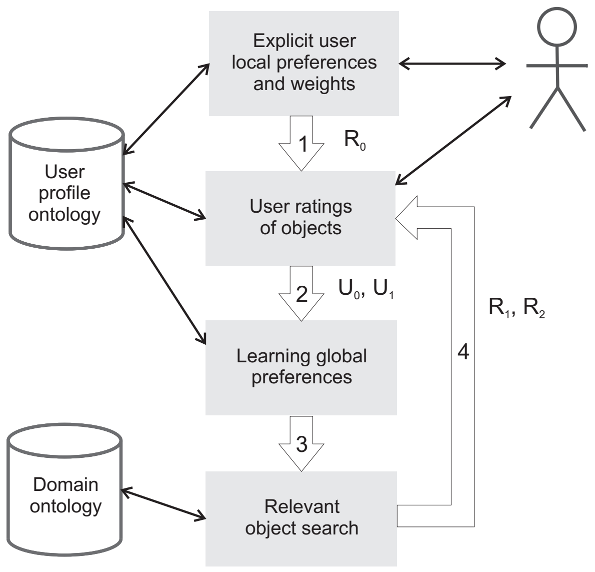


FIG. 3.1. *Flow diagram of user profile dependent part*

In our system and experiments, different users can explicitly specify local preferences. We consider job offer attributes like required education and experience level of the applicant, offered salary, manager position and the amount of traveling involved in the job (EducationLevel, ExperienceLevel, MinSalary, ManagementLevel, TravelingInvolved). The testing implementation first asks the user to specify his explicit preferences to these attributes. The user can choose if he prefers higher, lower, middle or marginal values of these attributes and he can even define how much he prefers every value (see Figure 3.2 and compare to Figure 2.1). Thus we can recognize *higher-best*, *lower-best*, *middle-best* and *marginal-best* preference types. Additionally a weight assigned to each attribute can be specified. This is used for combination function being weighted average. These parameters are used to retrieve (arrow 1) top-k objects to user. We denote this list of results as R_0.

User preferences can be changed or stated more precisely during several search cycles. We follow the idea that everybody can easily say, for a concrete object, how suitable it is. Thus we will require from user to evaluate the objects in scale from the worst to the best.

User evaluation uses 5 possible values ("worst", "bad", "neutral", "good", "best"). The k-tuple of initially retrieved objects R_0 is transformed to new set U_0 (a fuzzy set with different order induced by user ratings). This is sent (arrow 2) to our ILP tool learning global preferences. These are used (sent via arrow 3) to relevant object search to retrieve top-k objects from the whole set of objects, let us denote this by R_1. After new objects are given to user, he can evaluate them (creating U_1) and start the whole process again. We created a database log for user preferences, result sets and user ratings.

The testing involved 82 users who performed 333 cycles and rated 3547 results altogether. However, only 62 users passed at least one complete cycle successfully. The remaining users were "just browsing" or testing the system availability. In the following analysis we will consider only those users who performed at least one complete cycle, i. e. those who viewed and rated first 10 results and more precise global preferences were (possibly) found from their ratings.

Therefore we consider 62 users with 158 cycles. We find that global preferences were found from user ratings in 112 cycles, which is 71%. Successive cycles usually improve global preferences even further; the number of rules was higher or equal in 75% of successive cycles. Sometimes the number of rules oscillates from one cycle to another. This happens for 30% of users. We suppose that these users do not know exactly what they are looking for or their preferences do not remain the same throughout the testing. Although user's ratings are used for finding global preferences, they also provide us an important feedback. They define some ordering of results. If this ordering is similar to the ordering returned by top-k searching algorithm, it indicates that our preference model resembles real preferences well. We compare two orderings (the ordering R_i given by top-k algorithm and the ordering U_i given by user ratings) with Kendall tau rank correlation coefficient [11]. This coefficient determines a degree of correspondence between two orderings of the same objects.

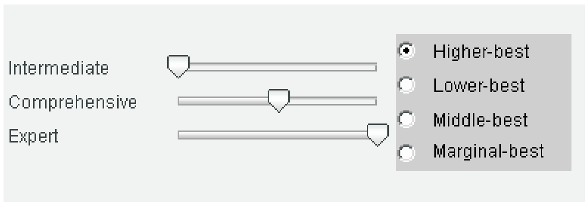


FIG. 3.2. *Graphical user interface for explicit preference specification*

Kendall tau coefficient is defined as $\tau = \frac{2P}{\frac{1}{2} \cdot n \cdot (n-1)} - 1$ where P is the number of concordant pairs in both ratings. If two orderings are the same, they have tau coefficient 1. If one ordering is the reverse of the other, they have tau coefficient -1.

R_0 and U_0 have average tau coefficient 0.44 representing a strong correlation. If we analyze second cycles (R_1 and U_1), we have an average tau coefficient 0.6; third cycles (R_2 and U_2) have 0.58. Note that some users stopped after first or second cycle, so the number of results is smaller for the third cycle. The correlation coefficients show that the inductive methods really improve global preferences. The decrease of tau coefficient in the third cycle is very small; it can be due to inconsistent user behavior.

We present a complete record of one typical user. Table 3.1 shows that this user rated the first 10 results and this ordering by ratings was different from the ordering given by top-k algorithm (tau correlation of R_0 and U_0 is only 0.1556). However, two rules were found from user's ratings and the global preferences were refined. The second result set R_1 has tau correlation 0.5111 compared to U_1. In the third cycle, the number of rules increased to 3 and tau correlation increased to 0.9111. The user was content with his results and he stopped searching.

We can also analyze the local preference types and find which type is the most common for every attribute (see Table 3.2). It is easy to see that *higher-best* is generally the most common type and marginal-best is the least common. Lower-best type is the second most frequent for ManagementLevel and TravelingInvolved, while middle-best is the second most frequent for EducationLevel, MinSalary and ExperienceLevel. These results reflect some intuitive ideas, e.g. that most people are not satisfied with low salary, seek a job for some specific required level of education or experience and that many of them are not willing to travel.

4. Conclusions.

In this paper we have described a model of system enabling users to search objects from the same domain and heterogeneous sources. Data are collected from various sources are processed to vector index and to a domain ontology. The system implements both user independent search and personalized search for users with different preferences. Our theoretical model integrates all parts of the system from collected data in classical RDF form to user query answering. We do not have a model for web resource downloading and ontology annotation part.

The model of user dependent search is based on modeling preferences by fuzzy sets and fuzzy logic. We present the semantics of fuzzy description logic f-$\mathcal{EL}$. User dependent search integrates both inductive and deductive approach. By induction we can learn local and global preferences (although currently we have implemented only global preferences). Results of induction as well as hand-filled orderings can be easily modeled in user ontology and fuzzy RDF. Deductive part of the system uses the preferences to find suitable objects for user, who can express his preferences precisely by fuzzy functions and aggregation function. The easiest way is to evaluate several objects in a scale and let the system to learn preferences. Repeating this process (evaluating a set of objects and finding new objects) leads to refining the user profile. User, who is satisfied with his profile and with the presented search results, does not have to evaluate objects again. When the domain ontology is actualized, user can just get the best objects according to his profile. Effective identification of suitable user type for a new user is the aim of our future research. It can be done by collecting more data from real users. We want to collect some personal information like age, gender, job position, etc. and the explicit specification of preferences from each user and then analyze the data in search for dependency.

Presented model was experimentally implemented and integration was tested using Cocoon and Spring Framework [14] and Corporate Memory of [5]. Both searching methods (user dependent and user independent search) are compound of tools which can be further improved separately. After the phase of gathering data about users, it will be possible to compare the efficiency, speed and complexity of these methods and decide about their practical usage. In future we plan to experiment with infrastructure of [18] and [3], which could possibly replace [14].

Our approach is used also in the Slovak project *NAZOU—Tools for acquisition, organization and maintenance of knowledge in an environment of heterogeneous information resources* [13] to find relevant job offers for the user according to his preferences.

TABLE 3.1
Testing Records for One User and Three Cycles

Offer ID	R_0	U_0	R_1	U_1	R_2	U_2	R_3
01096	2,837	1					
0f1b9	2,802	1					
01004	2,757	4					
01082	2,729	5					
01059	2,723	2					
01011	2,681	4					
9fe41	2,588	5	4	3			
8bc3a	2,588	4	4	4			
01090	2,54	5					
01095	2,54	2			3	2	
e4464			4	5	5	5	5
4c537			4	5	5	5	4
1d41d			4	1			
c56aa			4	5	5	4	4
01007			4	5	5	4	3
8f22d			4	5	5	3	3
01063			4	4	4	2	2
ef534			4	1			2
01069					3	2	
01068					3	1	
01100					3	1	
31a44							2
a8bbc							2
01036							2
τ	0,1556		0,5111		0,9111		
Rules		2		3		4	

TABLE 3.2
A Summary of Local Preference Types

Attribute	**Higher**	**Lower**	**Middle**	**Marginal**
EducationLevel	41	12	23	4
MinSalary	67	1	10	2
ExperienceLevel	50	10	15	5
ManagementLevel	43	24	8	5
TravelingInvolved	42	25	12	1

Similar strategy of communication with users (evaluation of sample objects) was used in [12], where the learning part of the system was covered by a neural network. This approach does not permit to model user preferences. We did not found any other similar approach to the whole process.

Acknowledgements. Partially supported by Czech projects 1ET100300419 and Slovak projects VEGA 1/3129/06 and NAZOU.

REFERENCES

[1] F. BAADER, R. KUESTERS, F. WOLTER, *Extensions to Description Logics*, in Description Logic Handbook, Cambridge University Press, 2003, pp. 219–261.

[2] P. BARTALOS, M. BARLA, G. FRIVOLT, M. TVAROŽEK, A. ANDREJKO, M. BIELIKOVÁ, P. NÁVRAT, *Building an Ontological Base for Experimental Evaluation of Semantic Web Applications*, in Proc. of SOFSEM 2007, Lecture Notes in Computer Science, Vol. 4362, Springer-Verlag, 2007, ISSN 0302-9743, pp. 682–692.

[3] D. BEDNÁREK, D. OBDRŽÁLEK, J. YAGHOB, F. ZAVORAL, *Data Integration Using DataPile Structure*, in Advances in Databases and Information Systems, Springer-Verlag, 2005, ISBN 3-540-42555-1, pp. 178–188.
[4] I. BRATKO, D. ŠUC, *Learning qualitative models*, AI Magazine 24 (2003), pp. 107-119.
[5] M. CIGLAN, M. BABIK, M. LACLAVÍK, I. BUDINSKÁ, L. HLUCHÝ, *Corporate memory: A framework for supporting tools for acquisition, organization and maintenance of information and knowledge*, in ISIM'06, Czech Republic, 2006, pp. 185–192.
[6] R. FAGIN, *Combining fuzzy information from multiple systems*, in J. Comput. System Sci., 58 (1999), pp. 83–99.
[7] R. FAGIN, A. LOTEM, M. NAOR, *Optimal Aggregation Algorithms for Middleware*, inProc. 20th ACM Symposium on Principles of Database Systems, 2001, pp. 102–113.
[8] P. GURSKÝ, *Towards better semantics in the multifeature querying*, in Proceedings of Dateso 2006, ISBN 80-248-1025-5, 2006, pp. 63–73.
[9] P. GURSKÝ, T. HORVÁTH, R. NOVOTNÝ, V. VANEKOVÁ, P. VOJTÁŠ, *UPRE: User preference based search system*, in Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI '06), Hong Kong, 2006, ISBN 0-7695-2747-7, pp. 841–844.
[10] T. HORVÁTH, P. VOJTÁŠ, *Induction of Fuzzy and Annotated Logic Programs*, in ILP 2006, LNAI 4455, Springer-Verlag, 2007, pp. 260–274.
[11] M. G. KENDALL, *Rank Correlation Methods*, Hafner Publishing Co., New York, 1955.
[12] E. NAITO, J. OZAWA, I. HAYASHI, N. WAKAMI, *A proposal of a fuzzy connective with learning function and query networks for fuzzy retrieval systems*, in Fuzziness in database management systems, P. Bosc and J. Kacprzyk, eds., Physica Verlag, 1995, pp. 345–364.
[13] *NAZOU. Tools for acquisition, organization and maintenance of knowledge in an environment of heterogeneous information resources*, [online, cited 11 November 2007], available from `http://nazou.fiit.stuba.sk`
[14] *Spring Framework. System for assembling components via configuration files*, [online, cited 11 November 2007], available from `http://www.springframework.org`
[15] A. SRINAVASAN, *The Aleph Manual*, Technical Report, Comp. Lab., Oxford University.
[16] P. VOJTÁŠ, *Fuzzy logic aggregation for Semantic Web search for the best (top-k) answers*, in Fuzzy logic and the semantic web, E. Sanchez, ed., Elsevier, 2006, pp. 341–360.
[17] P. VOJTÁŠ, *A fuzzy $\mathcal{EL}$ description logic with crisp roles and fuzzy aggregation for web consulting*, in Proc. IPMU'2006, B. Bouchon-Meunier et al., eds., EDK 2006, pp. 1834–1841.
[18] J. YAGHOB, F. ZAVORAL, *Semantic Web Infrastructure using DataPile*, in WI-IATW '06, Los Alamitos, California, ISBN 0-7695-2749-3, 2006, pp. 630–633.

Edited by: Maria Ganzha, Marcin Paprzycki
Received: Feb 4, 2008
Accepted: Feb 9, 2008

SCALABLE COMPUTING: PRACTICE AND EXPERIENCE
Volume 9, Number 1, pp. 61–68. http://www.scpe.org

ISSN 1895-1767
© 2008 SCPE

LOAD BALANCING FOR THE NUMERICAL SOLUTION OF THE NAVIER-STOKES EQUATIONS

GREGORY KARAGIORGOS* AND NIKOLAOS M. MISSIRLIS*

Abstract. In this paper we simulate the performance of a load balancing scheme. In particular, we study the application of the Extrapolated Diffusion (EDF) method for the efficient parallelization of a simple 'atmospheric' model. Our model involves the numerical solution of the steady state Navier-Stokes (NS) equations in the horizontal plane and random load values, corresponding to the "physics" computations, in the vertical plane. For the numerical solution of NS equations we use the Local Modified Successive Overrelaxation (LMSOR) method with local parameters thus avoiding the additional cost caused by the global communication of the involved parameter ω in the classical SOR method. We have implemented an efficient domain decomposition technique by using a larger number of processors in the areas of the domain with heavier work load. Our results show that in certain cases we have a gain as much as approximately 45% in execution time when our load balancing scheme is applied.

Key words. load balancing, domain decomposition, diffusion method.

1. Introduction. The efficient use of the available computational resources for running parallel applications in a distributed computing environment leads to the load balancing problem. Many parallel applications produce different workload during execution time. Parallel weather prediction simulations constitute typical applications that require load balancing techniques. However, the load has to be balanced in order to achieve an efficient use of the processor network. In such applications a geometric space is disrcetized by a 3d-grid. Then a domain decomposition technique assigns a subdomain to each processor of the network. Each processor performs the computations on mesh points in each subdomain independently. The involved computations in these grids are of two kinds: "dynamics" and "physics". The dynamics calculations correspond to the fluid dynamics of the atmosphere and are applied to the horizontal field. These calculations use explicit schemes to discretize the involved partial differential equations because they are inherently parallel. The physics calculations represent the natural procedures such as clouds, moist convection, the planetary boundary layer and surface processes and are applied to the vertical level. The computations of a vertical column are local, that is, they do not need data from the neighboring columns and are implicit in nature. As the "physics" computations differ from one subdomain to the other the processors have an unbalanced load. In such situations the load has to be balanced. Due to the nature of our problem we will consider iterative algorithms to balance the load. Most of the existing iterative load balancing schemes [12] involve two steps. The first step calculates a balancing flow. This flow is used in the second step, in which load balancing elements are migrated accordingly. It is well known that in distributed systems the communication complexity plays an important role. Therefore, a load balancing algorithm should have a minimum flow and use local communication. Iterative load balancing methods have these characteristics and this is the reason for making them appropriate in our case study. Two main categories are the Diffusion [1, 4], and the Dimension Exchange [4, 24] algorithms. The performance of a balancing algorithm can be measured in terms of number of iterations it requires to reach a balanced state and in terms of the amount of load moved over the edges of the underlying processor graph. In [6] it is shown that all local iterative diffusion schemes calculate a minimal flow with respect to a weighted l_2-norm. In the Diffusion (DF) method [4, 1] a processor simultaneously sends workload to its neighbors with lighter workload and receives from its neighbors with heavier workload. It is assumed that the system is synchronous, homogeneous and the network connections are of unbounded capacity. Under the synchronous assumption, the Diffusion (DF) method has been proved to converge in polynomial time for any initial workload [1].

In this paper we study the application of the DF method for the efficient parallelization of a simple model. Our model involves the numerical solution of the steady state Navier-Stokes (NS) equations in the horizontal plane and random load values, corresponding to the "physics" computations, in the vertical plane. Our intention is to study the performance of a load balancing scheme under more realistic conditions than in previous cases [13, 14]. Up to now the problem of load balancing has been studied solely, namely without any combination of horizontal and vertical computations. In this paper (i) we use an optimal version of the DF method and (ii) the migration of load transfer is implemented using only local communication as opposed to schedules which require global communication.

The paper is organized as follows. In Section 2 we present the Extrapolated Diffusion method, which possesses the fastest rate of convergence among its other counterparts [17]. In Section 3 we describe our simple 'atmospheric model' and we present the use of the Local Modified Successive Overrelaxation (LMSOR) method for the numerical solution of NS equations. In Section 4 we present a domain decomposition technique, which initially assigns a square domain to each processor in a mesh network. However, load balancing imposes a modified domain decomposition. As this increases

*Department of Informatics, University of Athens, Panepistimiopolis, 15784, Athens, Greece ({greg, nmis}@di.uoa.gr).

the communication complexity, we propose a load transfer to balance the load along the row and column processors only. This domain decomposition technique assigns a larger number of processors in the areas of the domain with heavier load, thus increasing the efficiency of load balancing. Finally, in Section 5 we present our simulation results.

2. The Extrapolated Diffusion method. Let us consider an arbitrary, undirected, connected graph $G = (V, E)$. This graph represents our processor network, where its nodes, namely the processors, are denoted by $v_i \in V$ and its edges (links) are $(v_i, v_j) \in E$, when $v_i,\ v_j$ are neighbors. Furthermore, we assign a weight $u_i \geq 0$ to each node, which corresponds to the computational load of v_i. The processor graph reflects the inter-connection of the subdomains of a mesh that has been partitioned and distributed amongst processors (see Figure 4.1). In this graph each node represents a subdomain and two nodes are linked by an edge if the corresponding subdomains share edges of the mesh.

The Extrapolated Diffusion (EDF) method for the load balancing has the form [1, 4]

$$u_i^{(n+1)} = u_i^{(n)} - \tau \sum_{j \in N(i)} c_{ij} \left(u_i^{(n)} - u_j^{(n)} \right), \tag{2.1}$$

where c_{ij} are diffusion parameters, $N(i)$ is the set of the nearest neighbors of node i of the graph $G = (V, E)$, $u_i^{(n)},\ i = 0, 1, 2, \ldots, |V|$ is the load after the n-th iteration on node i and $\tau \in \mathbb{R}\backslash\{0\}$ is a parameter that plays an important role in the convergence of the whole system to the equilibrium state. The overall workload distribution at step n, denoted by $u^{(n)}$, is the transpose of the vector $(u_1^{(n)}, u_2^{(n)}, \ldots, u_{|V|}^{(n)})$ and $u^{(0)}$ is the initial workload distribution. An alternative form of (2.1) is

$$u_i^{(n+1)} = (1 - \tau \sum_{j \in N(i)} c_{ij}) u_i^{(n)} + \tau \sum_{j \in N(i)} c_{ij} u_j^{(n)}, \tag{2.2}$$

which in matrix form becomes

$$u^{(n+1)} = M u^{(n)}, \tag{2.3}$$

where M is called the *diffusion matrix.* From (2.2) it follows that the elements of M, m_{ij}, are equal to τc_{ij}, if $j \in N(i)$, $1 - \tau \sum_{j \in N(i)} c_{ij}$, if $i = j$ and 0 otherwise. With this formulation, the features of diffusive load balancing are fully captured by the iterative process (2.3) governed by the diffusion matrix M. Also, (2.3) can be written as $u^{(n+1)} = (I - \tau L) u^{(n)}$, where $L = BWB^T$ is the *weighted Laplacian* matrix of the graph, W is a diagonal matrix of size $|E| \times |E|$ consisting of the coefficients c_{ij} and B is the vertex-edge incident matrix. At this point, we note that if $\tau = 1$, then we obtain the DF method proposed by Cybenko [4] and Boillat [1], independently. If $W = I$, i. e. if c_{ij}=constant, then we obtain the special case of the DF method with a single parameter τ *(unweighted Laplacian).* In the unweighted case and for network topologies such as chain, 2D-mesh, nD-mesh, ring, 2D-torus, nD-torus and nD-hypercube, optimal values for the parameter τ that maximize the convergence rate have been derived by Xu and Lau [23, 24]. Recently, we have solved the same problem in its most general form, the weighted case [17]. Next, we present a short review of our results as we use them to further accelerate the rate of convergence of DF.

The diffusion matrix of EDF can be written as

$$M = I - \tau L, \qquad L = D - A, \tag{2.4}$$

where $D = \mathrm{diag}(L)$ and A is the weighted adjacency matrix. Because of (2.4), (2.3) becomes $u^{(n+1)} = (I - \tau D)\, u^{(n)} + \tau A u^{(n)}$ which is the matrix form of (2.2).

The diffusion matrix M must have the following properties: nonnegative, symmetric and stochastic [4, 1]. The eigenvalues of L are $0 = \lambda_1 < \lambda_2 \leq \ldots \leq \lambda_{|V|}$. In case c_{ij} = constant, the optimum value of τ is attained at [22, 25]

$$\tau_o = \frac{2}{\lambda_2 + \lambda_{|V|}}$$

and the corresponding minimum value of the convergence factor

$$\gamma(M) = \max\{|1 - \tau \lambda_{|V|}|, |1 - \tau \lambda_2|\}$$

is given by

$$\gamma_o(M) = \frac{P(L) - 1}{P(L) + 1}, \text{ where } P(L) = \frac{\lambda_{|V|}}{\lambda_2},$$

TABLE 2.1
E=Even, O=Odd, Formulae for the optimum τ_o and $\gamma_o(M)$

N$_1$	N$_2$	τ_o	$\gamma_o(M)$
E	E	$\left(3 + 2\sigma_2 - \cos\frac{2\pi}{N_1}\right)^{-1}$	$\dfrac{1 + 2\sigma_2 + \cos\frac{2\pi}{N_1}}{3 + 2\sigma_2 - \cos\frac{2\pi}{N_1}}$
O	O	$\left(2 + \sigma_2(1 + \cos\dfrac{\pi}{N_2}) + \cos\dfrac{\pi}{N_1} - \cos\dfrac{2\pi}{N_1}\right)^{-1}$	$\dfrac{\cos\frac{\pi}{N_1} + \cos\frac{2\pi}{N_1} + \sigma_2(1 + \cos\frac{\pi}{N_2})}{2 + \sigma_2(1 + \cos\frac{\pi}{N_2}) + \cos\frac{\pi}{N_1} - \cos\frac{2\pi}{N_1}}$
E	O	$\left(3 - \cos\dfrac{2\pi}{N_1} + \sigma_2(1 + \cos\dfrac{\pi}{N_2})\right)^{-1}$	$\dfrac{1 + \cos\frac{2\pi}{N_1} + \sigma_2(1 + \cos\frac{\pi}{N_2})}{3 - \cos\frac{2\pi}{N_1} + \sigma_2(1 + \cos\frac{\pi}{N_2})}$
O	E	$\left(2 + 2\sigma_2 + \cos\dfrac{\pi}{N_1} - \cos\dfrac{2\pi}{N_1}\right)^{-1}$	$\dfrac{2\sigma_2 + \cos\frac{\pi}{N_1} - \cos\frac{2\pi}{N_1}}{2 + 2\sigma_2 + \cos\frac{\pi}{N_1} - \cos\frac{2\pi}{N_1}}$

which is the P-condition number of L. Note that if $P(L) \gg 1$, then the rate of convergence of the EDF method is given by

$$R(M) = -\log\gamma_o(M) \simeq \frac{2}{P(L)},$$

which implies that the rate of convergence of the EDF method is a decreasing function of $P(L)$. The problem of determining the diffusion parameters c_{ij} such that EDF attains its maximum rate of convergence is an active research area [5, 9, 17]. Introducing the set of parameters $\tau_i, i = 1, 2, \ldots, |V|$, instead of a fixed parameter τ in (2.2), the problem moves to the determination of the parameters τ_i in terms of c_{ij}. By considering local Fourier analysis [16, 17] we were able to determine good values (near the optimum) for τ_i. These values become optimum (see Table 2.1) in case the diffusion parameters are constant in each dimension and satisfy the relation $c_j^{(2)} = \sigma_2 c_i^{(1)}$, $i = 1, 2, \ldots, N_1$, $j = 1, 2, \ldots, N_2$, where $\sigma_2 = \frac{1-\cos\frac{2\pi}{N_1}}{1-\cos\frac{2\pi}{N_2}}$ and $c_i^{(1)}$, $c_j^{(2)}$ are the row and column diffusion parameters, respectively, of the torus [17]. Also, in [17] it is proved that for stretched torus, that is a torus with either $N_1 \gg N_2$ or $N_2 \gg N_1$,

$$R(EDF) \simeq 2R(DF),$$

at the optimum stage, which means that EDF is twice as fast as DF.

In order to further improve, by an order of magnitude, the rate of convergence of EDF we can apply accelerated techniques (Semi-Iterative, Second-Degree and Variable Extrapolation) following [22, 20, 15].

It is known [22, 25] that the convergence of (2.3) can be greatly accelerated if one uses the Semi-Iterative scheme

$$u^{(n+1)} = \rho_{n+1}(I - \tau_o L)u^{(n)} + (1 - \rho_{n+1})u^{(n-1)},$$

with

$$\rho_1 = 1,\ \rho_2 = \left(1 - \frac{\sigma^2}{2}\right)^{-1},\ \rho_{n+1} = \left(1 - \frac{\sigma^2}{4}\rho_n\right)^{-1},\ n = 2, 3, \ldots,$$

and

$$\sigma = \gamma_o(M). \tag{2.5}$$

It is worth noting that σ is equal to $\gamma_o(M)$, which is the minimum value of the convergence factor of EDF. In addition, $\gamma_o(M)$ and τ_o, for EDF, are given by the expressions of Table 2.1 for the corresponding values of N_1 and N_2. It can be shown [22, 25] that

$$R(SI - EDF) \simeq \sqrt{2}R(SI - DF)$$

which indicates that the rate of convergence of SI-EDF will be approximately 40% better than that of SI-DF in case of stretched torus.

3. The Atmospheric model. As a case study we will consider the simulation of a simple atmospheric model. In particular, we will consider the numerical solution of NS equations in the horizontal plane by using a parallel version of the SOR iterative method. In the vertical plane we model the implicit "physics" computations by random load values.

3.1. Discretization of the Navier-Stokes equations. The model problem considered here is that of solving the 2-D incopressible Navier-Stokes (NS) equations. The equations in terms of vorticity Ω and stream function Ψ are

$$\Delta\Psi = -\Omega, \qquad u\frac{\partial\Omega}{\partial x} + v\frac{\partial\Omega}{\partial y} = \frac{1}{Re}\Delta\Omega, \tag{3.1}$$

where Re is the Reynold number of the fluid flow and u, v are the velocity components in the y and x directions, respectively. The velocity components are given in terms of the stream function Ψ by $u = \frac{\partial\Psi}{\partial y}$ and $v = -\frac{\partial\Psi}{\partial x}$. If the computational domain is the unit square, then the boundary conditions for such flow are given by $\Psi = 0$, $\frac{\partial\Psi}{\partial x} = 0$ at $x = 0$ and $x = 1$, $\Psi = 0$, $\frac{\partial\Psi}{\partial y} = 0$ at $y = 0$ and $\Psi = 0$, $\frac{\partial\Psi}{\partial y} = -1$ at $y = 1$. The 5-point discretization of NS equations on a uniform $N \times N$ mesh of size $h = \frac{1}{N}$ leads to

$$\frac{1}{h^2}\left[-4\Psi_{ij} + \Psi_{i-1,j} + \Psi_{i+1,j} + \Psi_{i,j+1} + \Psi_{i,j-1}\right] = \Omega_{ij} \tag{3.2}$$

and

$$\Omega_{ij} = l_{ij}\Omega_{i-1,j} + r_{ij}\Omega_{i+1,j} + t_{ij}\Omega_{i,j+1} + b_{ij}\Omega_{i,j-1}, \tag{3.3}$$

with

$$l_{ij} = \frac{1}{4} + \frac{1}{16}Reu_{ij}, \; r_{ij} = \frac{1}{4} - \frac{1}{16}Reu_{ij},$$

$$t_{ij} = \frac{1}{4} - \frac{1}{16}Rev_{ij}, \; b_{ij} = \frac{1}{4} + \frac{1}{16}Rev_{ij},$$

where

$$u_{ij} = \Psi_{i,j+1} - \Psi_{i,j-1}, \; v_{ij} = \Psi_{i-1,j} - \Psi_{i+1,j}.$$

The boundary conditions are also given by

$$\Omega_{i,0} - \frac{2}{h^2}\Psi_{i,1} = 0, \; \Omega_{N,j} - \frac{2}{h^2}\Psi_{N-1,j} = 0$$

and

$$\Omega_{0,j} - \frac{2}{h^2}\Psi_{1,j} = 0, \; \Omega_{i,N} - \frac{2}{h^2}(h - \Psi_{i,N-1}) = 0.$$

3.2. The Local Modified SOR method. The local SOR method was introduced by Ehrlich [7, 8] and Botta and Veldman [2] in an attempt to further increase the rate of convergence of SOR. The idea is based on letting the relaxation factor ω vary from equation to equation. Kuo et. al [19] combined local SOR with red black ordering and showed that is suitable for parallel implementation on mesh connected processor arrays. In the present study we generalize local SOR by letting two different sets of parameters $\omega_{ij}, \omega_{ij}^{'}$ to be used for the red ($i+j$ even) and black ($i+j$ odd) points, respectively. An application of our method to (3.2) and (3.3) can be written as follows

$$\Omega_{ij}^{(n+1)} = (1 - \omega_{ij})\Omega_{ij}^{(n)} + \omega_{ij}J_{ij}\Omega_{ij}^{(n)}, \qquad i+j \text{ even}$$

$$\Omega_{ij}^{(n+1)} = (1 - \omega_{ij}^{'})\Omega_{ij}^{(n)} + \omega_{ij}^{'}J_{ij}\Omega_{ij}^{(n+1)}, \qquad i+j \text{ odd}$$

with

$$J_{ij}\Omega_{ij}^{(n)} = l_{ij}^{(n)}\Omega_{i-1,j}^{(n)} + r_{ij}^{(n)}\Omega_{i+1,j}^{(n)} + t_{ij}^{(n)}\Omega_{i,j+1}^{(n)} + b_{ij}^{(n)}\Omega_{i,j-1}^{(n)}$$

and

$$l_{ij}^{(n)} = \frac{1}{4} + \frac{1}{16} Reu_{ij}^{(n)},\ r_{ij}^{(n)} = \frac{1}{4} - \frac{1}{16} Reu_{ij}^{(n)},$$

$$t_{ij}^{(n)} = \frac{1}{4} - \frac{1}{16} Rev_{ij}^{(n)},\ b_{ij}^{(n)} = \frac{1}{4} + \frac{1}{16} Rev_{ij}^{(n)},$$

where

$$u_{ij}^{(n)} = \Psi_{i,j+1}^{(n)} - \Psi_{i,j-1}^{(n)}, v_{ij} = \Psi_{i-1,j}^{(n)} - \Psi_{i+1,j}^{(n)}.$$

A similar iterative scheme holds also for Ψ_{ij}. If μ_{ij} are real, then the optimum values of the LMSOR parameters are given by [3]

$$\omega_{1,i,j} = \frac{2}{1 - \overline{\mu}_{ij}\underline{\mu}_{ij} + \sqrt{(1-\overline{\mu}_{ij}^2)(1-\underline{\mu}_{ij}^2)}},\ \omega_{2,i,j} = \frac{2}{1 + \overline{\mu}_{ij}\underline{\mu}_{ij} + \sqrt{(1-\overline{\mu}_{ij}^2)(1-\underline{\mu}_{ij}^2)}},$$

where $\overline{\mu}_{ij}$ and $\underline{\mu}_{ij}$ are computed by

$$\overline{\mu}_{ij} = 2\left(\sqrt{\ell_{ij} r_{ij}} \cos \pi h + \sqrt{t_{ij} b_{ij}} \cos \pi k\right),$$

$$\underline{\mu}_{ij} = 2\left(\sqrt{\ell_{ij} r_{ij}} \cos \frac{\pi(1-h)}{2} + \sqrt{t_{ij} b_{ij}} \cos \frac{\pi(1-k)}{2}\right),$$

with $h = k = \frac{1}{\sqrt{N}}$. If μ_{ij} are imaginary, then the optimum values of the LMSOR parameters are given by

$$\omega_{1,i,j} = \frac{2}{1 - \overline{\mu}_{ij}\underline{\mu}_{ij} + \sqrt{(1+\overline{\mu}_{ij}^2)(1+\underline{\mu}_{ij}^2)}},\ \omega_{2,i,j} = \frac{2}{1 + \overline{\mu}_{ij}\underline{\mu}_{ij} + \sqrt{(1+\overline{\mu}_{ij}^2)(1+\underline{\mu}_{ij}^2)}}.$$

If μ_{ij} are complex of the form $\mu_{ij} = \mu_{R ij} + i\mu_{I ij}$, we use the following heuristic formulas [2]

$$\omega_{1,i,j} = \frac{2}{1 + \overline{\mu}_R \underline{\mu}_R - \overline{\mu}_I \underline{\mu}_I (1 - (\underline{\mu}_R \overline{\mu}_R)^{2/3})^{-1} + \sqrt{M_{R,I}}}$$

and

$$\omega_{2,i,j} = \frac{2}{1 - \overline{\mu}_R \underline{\mu}_R + \overline{\mu}_I \underline{\mu}_I (1 - (\underline{\mu}_R \overline{\mu}_R)^{2/3})^{-1} + \sqrt{M_{R,I}}},$$

where

$$M_{R,I} = [1 - \overline{\mu}_R^2 + \overline{\mu}_I^2 (1 - \overline{\mu}_R^{2/3})^{-1}][1 - \underline{\mu}_R^2 + \underline{\mu}_I^2 (1 - \underline{\mu}_R^{2/3})^{-1}].$$

4. Domain Decomposition and Load Transfer. Let us assume the domain for the solution of the NS equations is rectangular. Initially, we apply a domain decomposition technique which divides the original domain into p square subdomains, where p is the number of available processors (see Figure 4.1). This decomposition proved to be optimal, in case the load is the same on all mesh points [3], in the sense of minimizing the ratio communication over computation. Next, each subdomain is assigned to a processor in a mesh network. The parallel efficiency depends on two factors: an equal distribution of computational load on the processors and a small communication overhead achieved by minimizing the boundary length. If the latter is achieved by requiring the minimization of the number of cut edges i. e. the total interface length, the mesh partitioning problem turns out to be NP-complete. Fortunately, a number of graph partitioning heuristics have been developed (see e.g. [6, 10]). Most of them try to minimize the cut size which is sufficient for many applications but this approach has also its limitations [11]. To avoid these limitations we apply a load balancing scheme such as to maintain the structure of the original decomposition. This is achieved by adjusting, according to a simple averaging load rule, the width of each row/column. Although this approach cannot achieve full balance as it moves a constant number of columns and rows, it proves to be efficient. Next, we consider a load balancing scheme, which employs the above feature. Let us assume the situation as illustrated on the left of Figure 4.1. The shaded area denotes the physics computations, i. e. the load distribution. In an attempt to balance the load among the processors we decompose the load area into smaller domains as this is illustrated on the right of Figure 4.1. We will refer to this partitioning as "nesting". The advantage of nesting is that the structure of the domain decomposition graph remains unchanged, thus minimizing the interprocessor communication at the cost of imbalance. The problem now is to determine the width of each row and column. Let us consider the case presented in Figure 4.2, where we have four processors a, b, c, d, each one assigned initially a square with the same area. Further, we assume that the result of the EDF algorithm is that processor a must

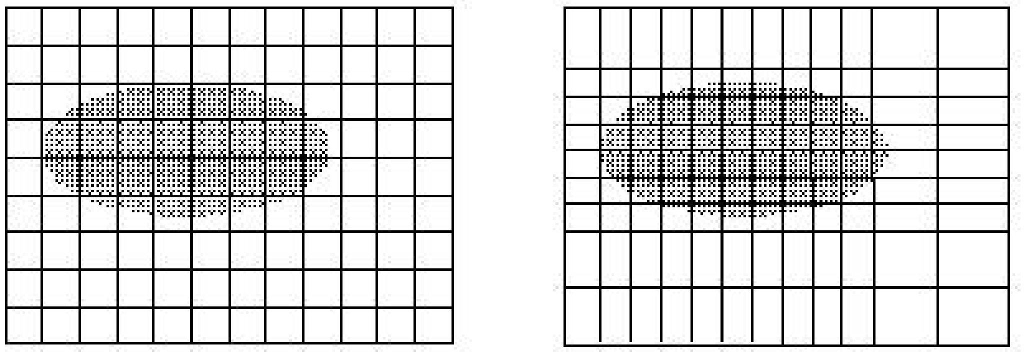


FIG. 4.1. *Domain decomposition by "nesting"*

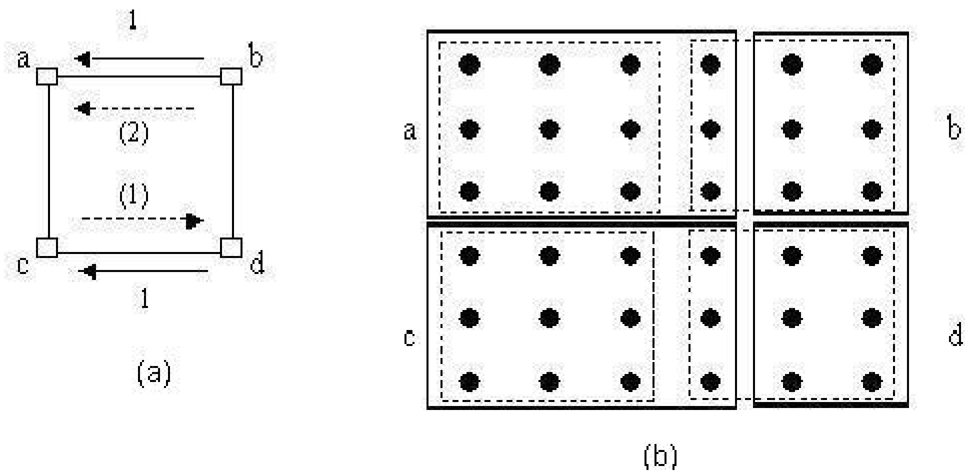


FIG. 4.2. *Column transfer according to the 'weighted average' method*

receive two columns of mesh points from processor b and processor c must send one column to processor d (Figure 4.2(a) dotted arrows). But if these transfers are carried out, they will destroy the mesh structure of the domain decomposition graph as now processor d will have two north neighbors a and b instead of one. In order to avoid this phenomenon we allow the 'weighted average' number of columns to be transferred among the processors a, b and c, d, where 'weighted average'= $\lceil \frac{2+(-1)}{2} \rceil = 1$. This means that processor a will receive one column (instead of two) and processor c will also receive one column (instead of sending one). After the load transfer is carried out, the domain graph remains a mesh as is depicted in Figure 4.2(b) with the solid lines, whereas the dotted lines indicate the initial boundaries of the processors. This process requires communication between processors along two successive rows of the mesh network of processors. A similar procedure for the processors along the columns will fix the width of each column. For a $\sqrt{p} \times \sqrt{p}$ mesh processor network this process requires a total of $O(\sqrt{p})$ communication.

5. Simulation Results. To evaluate the effectiveness of our load balancing algorithm, we ran some tests for the considered model using 44 processors of the HP Superdome parallel machine of the University of Athens. In these tests we compare the behavior of our model against the use of the load balancing algorithm. The method used for the load balancing was the SI-EDF [18]. For the physics computations we assumed a normal distribution of loads superimposed on the given mesh network for solving the NS equations. In particular, we considered that the artificial load is produced by the following function

$$f(x) = Me^{-(\frac{2(x-x_0)}{d})^2},$$

where x denotes the position of a processor in a row of the mesh network, M is the height and d is the width of the distribution (see figure 5.1). In our experiments we kept d constant. Clearly, by varying M we were able to examine the behavior of our load balancing algorithm in different scenarios between the "physics" and "dynamics" calculations.

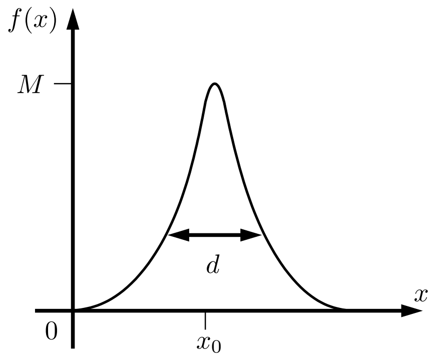


FIG. 5.1. *Random load values*

Our results are summarized in Table 5.1, where we considered different orders for the horizontal mesh as shown in the second column. For each mesh we find the improvement achieved by using our load balancing scheme. The numbers in the 3rd and 4th columns are in secs. By increasing the order of the mesh the rows and columns for each subdomain are also increased. As a consequence, the computation and communication time of LMSOR is also increased. This is clearly observed in both columns (With LB, Without LB) of Table 5.1. Increasing M has the effect of increasing the load in the normal distribution. For small M ($M = 1$) the improvement, by using load balancing, is constant and approximately 21% for meshes with order larger than 40. This was expected as for light load there will be a small number of row/column movement which will be approximately the same for sufficiently large meshes. Also, the overall gain is moderate. For small meshes the horizontal computation time is not large enough to justify the use of load balancing resulting in negative improvement. For heavier load ($M = 100$) the gain in time, due to load balancing, will be greater as is shown in Table 5.1, where we have an improvement of approximately 45% for larger meshes. However, if the load is increased considerably there will also be an increase in load balancing time as more columns and rows need to be moved according to our load balancing algorithm. This has the effect of reducing the improvement. Indeed, if $M = 1000$, then the improvement is less than 45%. From the above it is clear that when the vertical load is small in comparison to the horizontal then load balancing should be avoided. On the other hand, as the load increases, we may reach an improvement of nearly 45% when using the load balancing algorithm. Even though we kept the structure of the application graph unchanged at the cost of approximate balance, the gain is satisfactory.

6. Acknowledgement. The project is co-funded by the European Social Fund and National Resources (EPEAK II) Pythagoras, Grant No. 70/3/7418.

REFERENCES

[1] Boillat J. E., *Load balancing and poisson equation in a graph* , Concurrency: Practice and Experience 2, 1990, 289–313.
[2] Botta E. F. and Veldman A. E. P., *On local relaxation methods and their application to convection-diffusion equations*, J. Comput. Phys., 48, 1981, 127–149.
[3] Boukas L.A. and Missirlis N. M., *The Parallel Local Modified SOR for nonsymmetric linear systems*, Inter. J. Computer Math., 68, 1998, 153–174.
[4] Cybenko G., *Dynamic load balancing for distributed memory multi-processors*, Journal of Parallel and Distributed Computing, 7, 1989, 279–301.
[5] Diekmann R., Muthukrishnan S. and Nayakkankuppam M. V., *Engineering diffusive load balancing algorithms using experiments*. In G. Bilardi et al., editor, IRREGULAR'97, LNCS 1253, 1997, 111–122.

TABLE 5.1
Simulation Results. $\mathbf{LB}$ *=Load Balancing. Numbers in 3nd and 4th columns are in secs. The last column shows the percentage in improvement.*

Scale Factor	**Mesh Order**	**44 Processors**		
		Without LB	**With LB**	**Improvement**
$M = 1$	20×20	46.932	50.551	-7.71%
	40×40	149.88	117.306	21.73%
	80×80	366.119	287.374	21.51%
	100×100	499.268	394.327	21.02%
$M = 100$	20×20	4689.879	4652.724	0.79%
	40×40	15517.46	9654.508	37.78%
	80×80	37866.911	20745.52	45.21%
	100×100	48499.712	26434.305	45.50%
$M = 1000$	20×20	489042.981	461727.89	5.59%
	40×40	1499122.957	1112865.135	25.77%
	80×80	3767658.518	2186112.869	41.98%
	100×100	4700380.338	2834764.494	30.69%

[6] Diekmann R., Frommer A. and Monien B., *Efficient schemes for nearest neighbour load balancing*, Parallel Computing, 25, 1999, 789–812.
[7] Ehrlich L. W., *An Ad-Hoc SOR Method*, J. Comput. Phys., 42, 1981, 31–45.
[8] Ehrlich L. W., *The Ad-Hoc SOR method: A local relaxation scheme*, in elliptic Problem Solvers II, Academic Press, New York, 1984, 257–269.
[9] Elsässer R., Monien B., Schamberger S. and Rote G., *Optimal Diffusion Schemes and Load Balancing for Product Graphs* , Parallel Proc. Letters, 14, No.1, 2002, 61–73.
[10] Farhat C., Maman N. and Brown G., *Mesh partitioning for implicit computations via iterative domain decomposition: Impact and optimization of the subdomain aspect ratio*, Int. J. Numer. Meth. in Eng., 38, 1995, 989–1000.
[11] Hendrickson B. and Leland R., *An improved spectral graph partitioning algorithm for mapping parallel computations*, SIAM J. Sci. Comput., 16, 1995, 452–469.
[12] Hu Y. F. and Blake R. J., *An improved diffusion algorithm for dynamic load balancing*, Parallel Computing, 25, 1999, 417–444.
[13] Karagiorgos G., Missirlis N. M. and Tzaferis F., *Dynamic Load Balancing for Atmospheric Models*, Proceedings of the Ninth ECMWF Workshop on the use of High Performance Computing in Meteorology, Developments in teracomputing, November 13-17, 2000, Reading, UK, edit. W. Zwieflhofer, N. Kreitz, World Scientific, 214–226.
[14] Karagiorgos G. and Missirlis N. M., *Iterative Load Balancing Schemes for Air Pollution Models in Large-Scale Scientific Computing*, Lecture Notes in Computer Science 2179, S. Margenov, J. Wasniewski and P. Yalamov, Third International Conference, Sozopol, Bulgaria, 2001, 291–298.
[15] Karagiorgos G. and Missirlis N. M., *Accelerated diffusion algorithms for dynamic load balancing*, Information Processing Letters, 84, 2002, 61–67.
[16] Karagiorgos G. and Missirlis N. M., *Fourier analysis for solving the load balancing problem*, Foundations of Computing and Decision Sciences, 27, No 3, 2002, 129–140.
[17] Karagiorgos G. and Missirlis N. M., *Convergence of the diffusion method for weighted torus graphs using Fourier analysis*, Theoretical Computer Science (accepted).
[18] Karagiorgos G., Missirlis N. M. and Tzaferis F., *Fast diffusion load balancing algorithms on torus graphs*, Proceedings of EuroPar 06, Dresden, Germany, August 2006, 222–231.
[19] Kuo C.-C. J., Levy B. C. and Musicus B. R., *A local relaxation method for solving elliptic PDE's on mesh-connected arrays*, SIAM J. Sci. Statist. Comput. 8, 1987, 530–573.
[20] Muthukrishnan S. , Ghosh B. and Schultz M. H. , *First and second order Diffusive methods for rapid, coarse, distributed load balancing*, Theory of Computing Systems, 31, 1998, 331–354.
[21] Ostromsky Tz. and Zlatev Z., *Parallel Implementation of a Large-Scale 3-D Air Pollution Model*, Lecture Notes in Computer Science 2179, S. Margenov, J. Wasniewski and P. Yalamov, Third International Conference, Sozopol, Bulgaria, 2001, 309–316.
[22] Varga R. S., *Matrix Iterative Analysis*, Englewood Cliffs, NJ, Prentice-Hall, 1962.
[23] Xu C. Z. and Lau F. C. M., *Optimal parameters for load balancing the diffusion method in k-ary n-cube networks*, Information Processing Letters 4, 1993, 181–187.
[24] Xu C. Z. and Lau F. C. M., *Load balancing in parallel computers: Theory and Practice*, Kluwer Academic Publishers, Dordrecht, 1997.
[25] Young D. M., *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.

Edited by: Svetozar Margenov
Received: Sept 17, 2007
Accepted: Oct 29, 2007

SCALABLE COMPUTING: PRACTICE AND EXPERIENCE
Volume 9, Number 1, pp. 69–75. http://www.scpe.org

ISSN 1895-1767
© 2008 SCPE

AN MIMD ALGORITHM FOR FINDING MAXIMUM AND MINIMUM ON OMTSE ARCHITECTURE

S. C. PANIGRAHI* AND G. SAHOO*

Abstract. Optical Multi-Trees with Shuffle Exchange (OMTSE), recently proposed, is an efficient model of optoelectronic parallel computer. The OMTSE interconnection system consists of n^2 factor networks called TSE networks, which are organized in the form of an $n \times n$ matrix. Each factor network has n number of leaf nodes. The network has a total of $3n^3/2$ nodes. The diameter and bisection width of the network is $6 \log n - 1$ and $n^3/4$ respectively. In this paper we present a synchronous MIMD algorithm to find the maximum and minimum of $n(n-1)^2$ data elements on OMTSE with $(4 \log n + 4)$ electronic moves and 3 optical moves.

Key words. parallel processing, time complexity, optical multi-trees with shuffle exchange, OTIS mesh, optoelectronic computers.

1. Introduction. The interconnection network is the heart of a parallel processing system, and many systems have failed to meet their design goals for the design of their essential components. The performance of most digital systems today is limited by their interconnection network bandwidth. The bandwidth limitation of the electronic interconnects prompted the need for exploring alternatives that overcome this limitation. Optics is considered as an alternative that is capable of providing inherent communication, parallelism, high connectivity and large bandwidth. It is well known that when the communication distances exceed few millimeters, optical interconnects provide advantage over the electronic interconnects in term of power, speed and crosstalk property [1, 10]. Therefore, in the construction of very powerful and large multiprocessor systems, it is advantageous to interconnect close processors (with in same physical package, e.g. chip) physically using electronic links and far processors (kept in other package) using optical links. Motivated by these observations new hybrid computer architecture utilizing both optical and electronic technologies have been proposed and investigated in [9, 8].

Marsden et al. [9], Hendrick et al. [14] and Zane et al. [8] have proposed an architecture in which the processors are partitioned into groups where processors within each group are connected by electronic links, while those in different groups are connected using optical interconnections. The Optical Transpose Interconnect System (OTIS) proposed by Marsden et al. [9] is an example of such hybrid architecture in which processors are partitioned into groups of same size, and processor i of group j is connected to the processor j of group i via an optical link. Krishnamoorthy et al. [1] have shown that when number of processors in each group is equal to the total number of groups, then bandwidth and power efficiency are maximized, and system area and volume are minimized.

The OTIS mesh optoelectronic computer is a class of OTIS computers where processor in each group are interconnected by electronic links followings the two-dimensional mesh paradigm. An N-processor OTIS-Mesh [4] has a diameter of $4N^{1/4} - 3$. Mapping algorithms of various fundamental problems occurring in real-life applications on the OTIS-Mesh has been studied by several authors, e.g., Wang and Sahni [4, 5, 6, 7], Rajasekarna and Sahni [13], Osterloh [1].

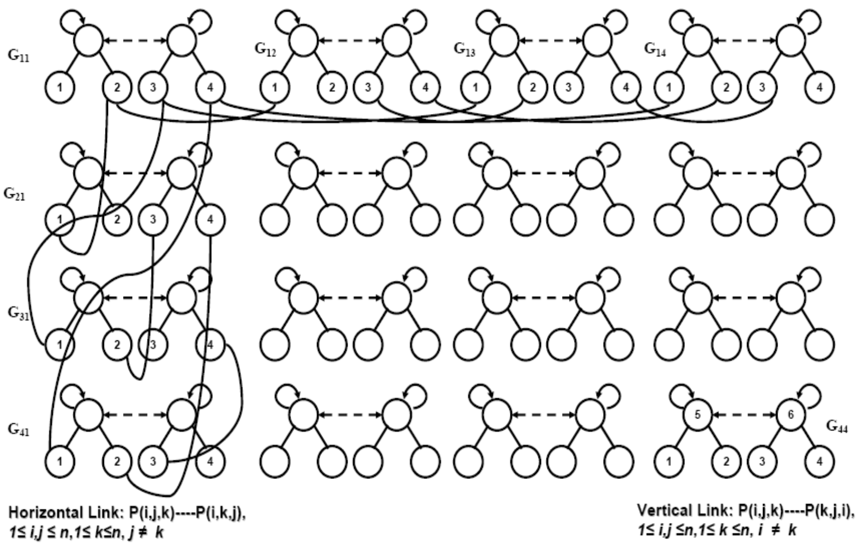
Optical Multi-Trees with Shuffle Exchange (OMTSE) is a new interconnection network proposed by Panigrahi et al. [11] for optoelectronic parallel computers. The network consists of a total of $3n^3/2$ processors are built around n^2 factor networks called TSE networks. Each factor network consists of n leaf nodes. The diameter and bisection width of the OMTSE network is shown to be $6 \log n - 1$ and $n^3/4$. Basic broadcast operations and several parallel algorithms including summation, prefix computation, matrix transpose, matrix multiplication, sorting have shown to map on OMTSE [11, 12] in lesser time than the OMULT [3].

In this paper we have introduced a synchronous MIMD algorithm for finding maximum and minimum of m elements on OMTSE where $m = n(n-1)^2$. All processors of OMTSE interconnection system operate in a lock step fashion. The algorithm has shown to run in $(4 \log n + 4)$ electronic moves and 3 optical moves. For the purpose of the complexity analysis of our algorithm, we count the data moves along the electronic links (known as electronic moves) and optical links (known as optical moves) separately.

The rest of the paper is organized as follows. The topology of the network is described in section 2. The proposed algorithm is described in the section 3 followed by the feasibility analysis and conclusion presented in section 4 and section 5 respectively.

2. Topology of OMTSE. The factor network used in OMTSE topology is a two layer TSE (Trees with Shuffle Exchange) network, which is nothing but an interconnection network containing a group of 2^r, $r \geq 1$, complete binary trees of height one and the roots of these binary trees are connected with Shuffle-Exchange fashion. The proposed OMTSE

*Deaprtment of Computer Science and Engineering, Birla Institute of Technology, Mesra, Ranchi 835215, India (satish.panigrahi@gmail.com, gsahoo@bitmesra.ac.in).



G11
G12
G13
G14
G21
G31
G41
G44
Horizontal Link: P(i,j,k)----P(i,k,j), 1≤ i,j ≤ n, 1≤ k≤n, j ≠ k
Vertical Link: P(i,j,k)----P(k,j,i), 1≤ i,j ≤n, 1≤ k ≤n, i ≠ k

FIG. 2.1. *An example of OMTSE topology with $n = 4$.*

interconnection system consists of n^2 TSE networks, which are organized in the form of an $n \times n$ matrix. We denote the TSE network placed at i^{th} row and j^{th} column of this matrix by G_{ij}, $1 \le i, j \le n$. Each TSE network having n nodes at layer 2 and $n/2$ nodes at layer 1. There are, therefore, $N = 3n^3/2$ processors in total. The nodes within each TSE network are interconnected by usual electronic links, while the nodes at layer 2 (i. e. leaf processors) of different TSE networks are interconnected by optical links according to the rules given below. Let us label the nodes in each TSE network G_{ij}, $1 \le i, j \le n$, by distinct integers from 1 to $3n/2$ in reverse order, i. e., the nodes at layer 2 of TSE network are numbered from 1 to n in order from left to right, and nodes at layer 1 also numbered from left to right. The node k in a TSE network G_{ij} will be referred to by the processor $P(i, j, k)$, $1 \le i, j \le n$, $1 \le k \le 3n/2$. The optical links interconnecting only the leaf nodes in different TSE networks are used in the following way

1. Processor $P(i, j, k)$, $1 \le i, j \le n$, $1 \le k \le n$, $j \ne k$, is connected to the processor $P(i, k, j)$ by bidirectional optical link called horizontal inter-TSE link.
2. Processor $P(i, j, k)$, $1 \le i, j \le n$, $1 \le k \le n$, $i \ne k$, is connected to the processor $P(k, j, i)$ by bidirectional optical link called vertical inter-TSE link.

The diameter of a network is the maximum distance between any two processing nodes in the network. Hence starting from a node $P(i, j, k)$, $1 \le i, j \le n$, $1 \le k \le n$ we can reach another node $P(i', j', k')$, $1 \le i', j' \le n$, $1 \le k' \le n$ of the OMTSE interconnection system by traversing the following path

$$P(i, j, k) \to P(i, j, j') \to P(i, j', j) \to P(i, j', i') \to P(i', j', i) \to P(i', j', k')$$

It can easily be seen that the diameter of OMTSE topology is $6 \log n - 1 = O(\log n)$, with $6 \log n - 3$ electronic links and 2 optical links. Similarly, we can find out the bisection width of OMTSE topology which is equal to $n^3/4$. An example of OMTSE topology for $n = 4$ with partial links is shown in Fig. 2.1.

3. Proposed Algorithm. We assume that each processor $P(i, j, 1)$, $1 \le i, j \le n$ has two registers $A(i, j, 1)$ and $B(i, j, 1)$, where as the rest of the processors has only one register $A(i, j, k)$, $1 \le i, j \le n$, $2 \le k \le 3n/2$. Suppose we have $m = n(n-1)^2$ data elements stored in A-register of n leaf nodes of all TSE networks excluding the TSE networks in the first row and first column of OMTSE. Hence each factor networks excluding the factor networks in the first row and in the first column of OMTSE contain n data elements in its leaf nodes. We can find the maximum and minimum of these elements in $(4 \log n + 4)$ electronic moves and 3 optical moves. The steps of the proposed algorithm have been described below, where we refer the left child and right child of the processor $P(i, j, k)$ as LCHILD$(P(i, j, k))$ and RCHILD$(P(i, j, k))$ respectively.

Algorithm: (ALGO_MAX_MIN). Initial Condition: The m elements stored in A-registers of $P(i, j, k)$, $2 \le i, j \le n$, $1 \le k \le n$ and initialize $A(i, 1, 1)$ as zero for $1 \le i \le n$, and $A(1, j, 1)$ as ∞ for $2 \le j \le n$.

```
1. For all $i, j, k$, $2 \le i, j \le n$ and $n+1 \le k \le 3n/2$, do in parallel $A(i, j, k) \leftarrow \max\big(A(LCHILD(P(i, j, k))) + A(RCHILD(P(i, j, k)))\big)$
```

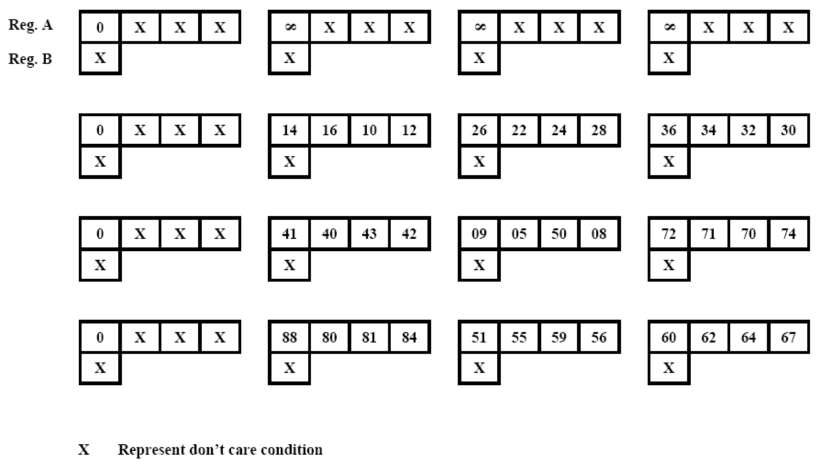


FIG. 3.1. *Initial Distribution of Data Elements.*

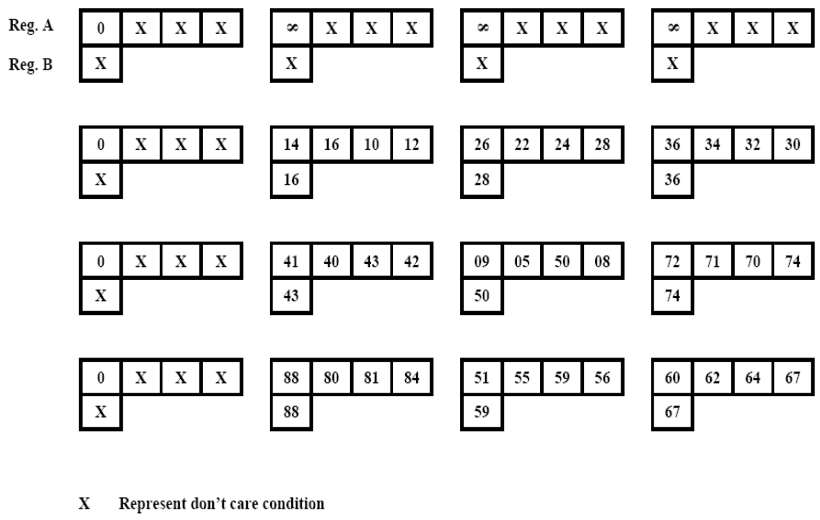


FIG. 3.2. *Data Distribution after step 3.*

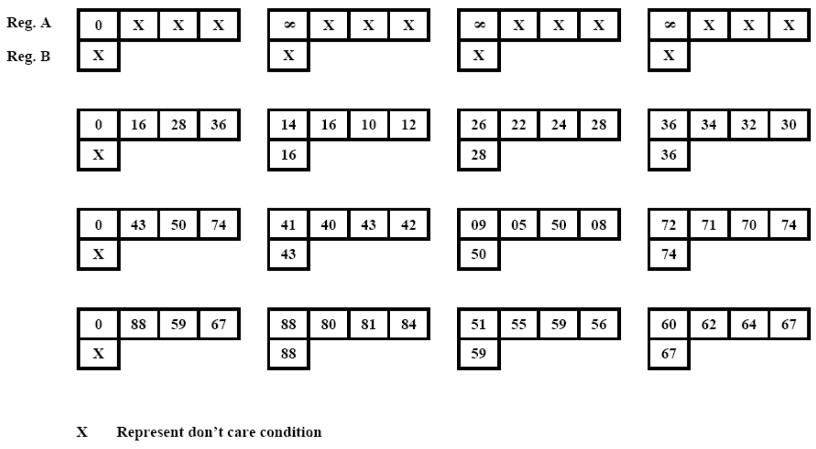


FIG. 3.3. *Data Distribution after step 4.*

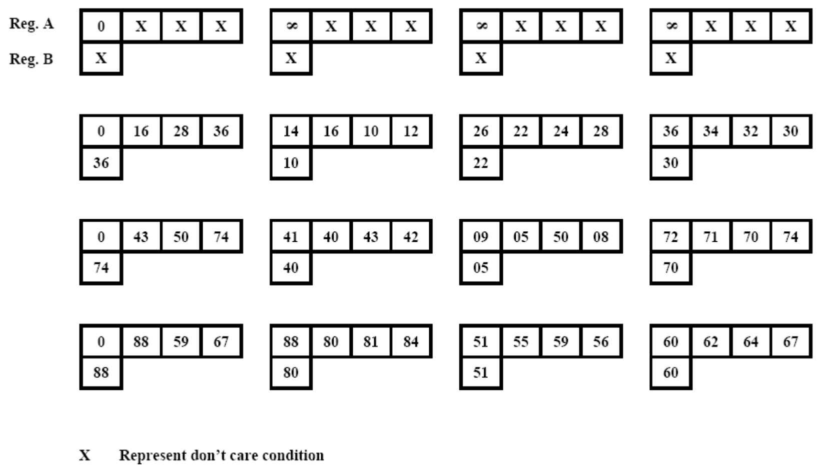


FIG. 3.4. *Data Distribution after step 7.*

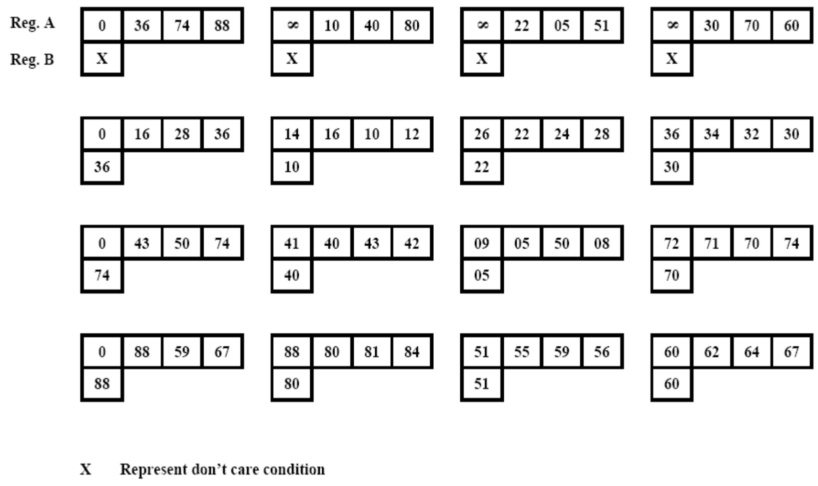


FIG. 3.5. *Data Distribution after step 8.*

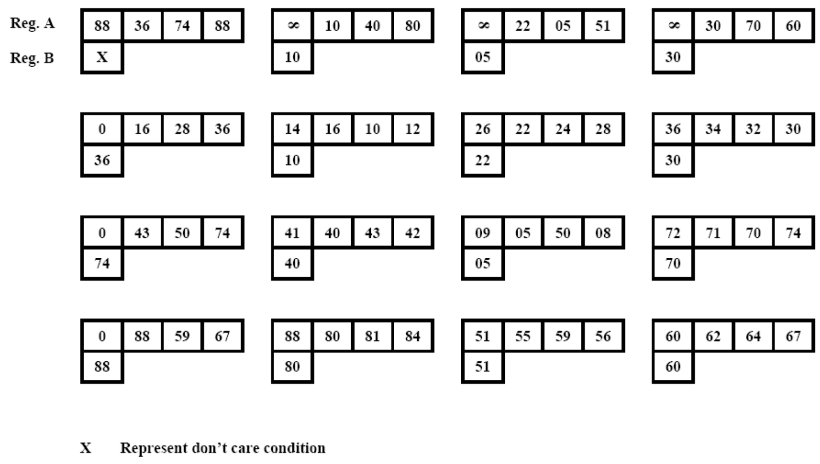


FIG. 3.6. *Data Distribution after step 11.*

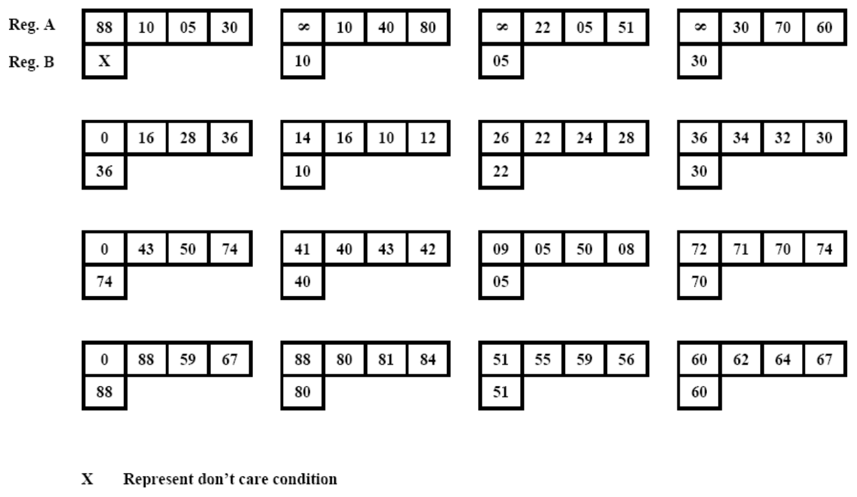


FIG. 3.7. *Data Distribution after step 12.*

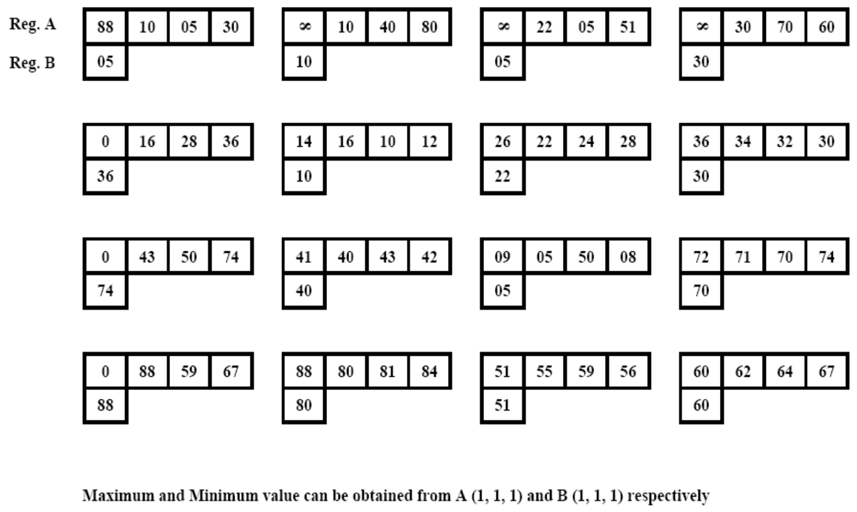


FIG. 3.8. *Maximum and Minimum value can be obtained from* $A(1,1,1)$ *and* $B(1,1,1)$.

```
2. Find the maximum from the data stored in layer 1 of all TSE networks, $G_{ij}$, $2 \leq i, j \leq n$, of OMTSE interconnection system.
3. For all $i, j$, $2 \leq i, j \leq n$;
   $B(i,j,1) \leftarrow A(i,j,n+1)$
4. For all $i, j$, $2 \leq i, j \leq n$;
   $A(i,1,j) \leftarrow B(i,j,1)$ /* horizontal inter-TSE link */
5. do in parallel
   (i) For all $i, k$; $2 \leq i \leq n$ and $n+1 \leq k \leq 3n/2$, do in parallel
       $A(i,1,k) \leftarrow \max\big(A(LCHILD(P(i,1,k))) + A(RCHILD(P(i,1,k)))\big)$
   (ii) For all $i, j, k$, $2 \leq i, j \leq n$ and $n+1 \leq k \leq 3n/2$, do in parallel
       $A(i,j,k) \leftarrow \min\big(A(LCHILD(P(i,j,k))) + A(RCHILD(P(i,j,k)))\big)$
6. do in parallel
   (i) Find the maximum from the data stored in layer 1 of all TSE networks, $G_{i1}$, $2 \leq i \leq n$, of OMTSE interconnection system.
   (ii) Find the minimum from the data stored in layer 1 of all TSE networks, $G_{ij}$, $2 \leq i, j \leq n$, of OMTSE interconnection system.
```

```
 7. For all $i, j, 2 \le i \le n, 1 \le j \le n$, do in parallel
    $B(i, j, 1) \leftarrow A(i, j, n+1)$
 8. For all $i, j, 2 \le i \le n, 1 \le j \le n$, do in parallel
    $A(1, j, i) \leftarrow B(i, j, 1)$ /* vertical inter-TSE link */
 9. do in parallel
     (i) For all $k$; $n+1 \le k \le 3n/2$, do in parallel
         $A(1,1,k) \leftarrow \max\big(A(LCHILD(P(1,1,k))) + (RCHILD(P(1,1,k)))\big)$
    (ii) For all $j, k$; $2 \le j \le n$ and $n+1 \le k \le 3n/2$, do in parallel
         $A(1,j,k) \leftarrow \min\big(A(LCHILD(P(1,j,k))) + A(RCHILD(P(1,j,k)))\big)$
10. do in parallel
     (i) Find the maximum from the data stored in layer 1 of all TSE networks, $G_{11}$ of OMTSE interconnection
         system.
    (ii) Find the minimum from the data stored in layer 1 of all TSE networks, $G_{1j}$, $2 \le j \le n$, of OMTSE
         interconnection system.
11. do in parallel
     (i) $A(1,1,1) \leftarrow A(1,1,n+1)$
    (ii) For all $j$, $2 \le j \le n$,
         $B(1,j,1) \leftarrow A(1,j,n+1)$
12. For all $j$, $2 \le j \le n$,
    $A(1,1,j) \leftarrow B(1,j,1)$ /* horizontal inter-TSE link */
13. For all $k$; $n+1 \le k \le 3n/2$, do in parallel
    $A(1,1,k) \leftarrow \min\big(A(LCHILD(P(1,1,k))) + (RCHILD(P(1,1,k)))\big)$
14. Find the minimum from the data stored in layer 1 of all TSE networks, $G_{11}$ of OMTSE interconnection system.
15. $B(1,1,1) \leftarrow A(1,1,n+1)$.
```

The above algorithm has been illustrated through Fig. 3.1 to 3.8, for m distinct positive numbers, all of which are less than a given number (say, ∞). The output can be taken respectively form $A(1,1,1)$ and $B(1,1,1)$ for maximum and minimum of m elements. Steps 4, 8 and 12 require only one move through optical link each, step 2, 6, 10, and 14 require $(\log n - 1)$ data moves through electronic link where as rest of the instructions require one move through each electronic link. The overall time of the computation is $(4\log n + 4)$ electronic links and 3 optical links.

The comparative analysis of the above algorithm with the corresponding SIMD algorithm [12] in respect to the size of data set is given in the following section.

4. Feasibility Analysis. The proposed MIMD algorithm for finding the maximum and minimum of m data elements on OMTSE requires an overall $4\log n + 7$ number of moves where as the SIMD approach require $6\log n + 10$ for n^3 data elements [12]. In both cases the algorithm uses a total of $3n^3/2$ processors, built around n^2 TSE networks of OMTSE interconnection system. Taking the number of data elements into consideration, the study of feasibility of the proposed algorithm has been done with a total number of $n^4(n-1)^2$ data elements. For these number of data elements it can be seen that SIMD and MIMD algorithms can be repeated to a maximum of $n^3 - 2n^2 + n + 2$ and $n^3 + 4$ times respectively. Consequently the total number of data movement (both optical and electronic) can respectively be obtained as $\lambda(n) = (6\log n + 10)(n(n-1)^2 + 2)$ and $\mu(n) = (4\log n + 7)(n^3 + 4)$. For various values of n, $\lambda(n)$ and $\mu(n)$ are depicted in Fig. 4.1.

5. Conclusion. This paper introduces a synchronous MIMD algorithm for finding both maximum and minimum of a set of data elements on OMTSE interconnection system with $O(\log n)$ time. It turns out from above discussion that the overall performance of the proposed algorithm is much more satisfactory than the existing SIMD algorithm discussed in [12].

REFERENCES

[1] A. KRISHNAMOORTHY, P. MARCHAND, F. KIAMILEV AND S. ESNER, *Grain-Size Considerations for Optoelectronic Multistage Interconnection Networks*, Applied Optics, Vol. 31, no. 26, 1992, pp. 5480–5507.

[2] A. OSTERLOH, *Sorting on the OTIS-Mesh*, Proc. 14th International Parallel and Distributed Processing Symposium (IPDPS 2000), 2000, pp. 269–274.

[3] B. P. SINHA, S. BANDYOPADHYAY, *OMULT: An Optical Interconnection System for Parallel Computing*, Proc. of International Conference on Computers and Devices for Communications, 2004.

[4] C. -F. WANG, S. SAHNI, *Basic operations on the OTIS-Mesh optoelectronic computer*, IEEE Transaction on Parallel and Distributed Systems, Vol. 9, no.12, 1998, pp. 1226–1236.

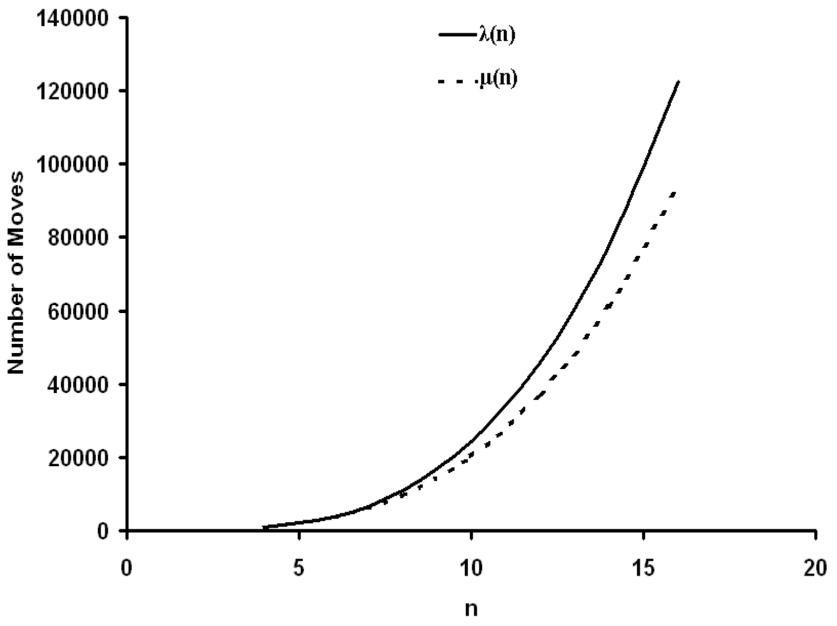


FIG. 4.1. *Comparison between $\lambda(n)$ and $\mu(n)$.*

[5] C. -F. WANG, S. SAHNI, *BPC permutations on the OTIS-Mesh optoelectronic computer*, Proc. of the International Conference on Massively Parallel processing using Optical Interconnections (MPPOI), 1997, pp. 130–135.

[6] C. -F. WANG, S. SAHNI, *Image Processing on the OTIS-Mesh optoelectronic computer*, IEEE Transaction on Parallel and Distributed Systems, Vol. 11, no.2, 2000, pp. 97–109.

[7] C. -F. WANG, S. SAHNI, *Matrix Multiplication on the OTIS-Mesh optoelectronic computer*, IEEE Transaction on Computers, Vol. 50, No.7, 2001, pp. 635–646.

[8] F. ZANE, P. MARCHAND, R. PATURI, S. ESENER, *Scalable Network Architectures Using Optical Transpose Interconnection System (OTIS)*, Journal of Parallel and Distributed Computing, Vol. 60, no. 5, 2000, pp. 521–538.

[9] G. C. MARSDEN, P. J. MARCAAND, P. HARVEY, S. C. ESNER, *Optical Transpose Interconnection System Architectures*, Optics Letters, Vol. 18, no.13, 1993, pp. 1083–1085.

[10] M. FELDMAN, S. ESENER, C. GUEST, S. LEE, *Comparison between Electrical and Free-Space Optical Interconnects Based on Power and Speed Consideration*, Applied Optics, Vol. 27, no.9, 1988, pp. 1742–1751.

[11] S. C. PANIGRAHI, S. PAUL, G. SAHOO, *OMTSE - An Optical Interconnection System for Parallel Computing*, Proc. 14th International Conference on Advanced Computing and Communication (ADCOM), 2006, pp. 626–627.

[12] S. C. PANIGRAHI, S. PAUL, G. SAHOO, *Parallel Prefix Computation, Sorting and Reduction Operation on OMTSE architecture*,Proc. of the International Conference on Advanced Computing and Communications (ICACC), 2007, pp. 616–620.

[13] S. RAJASEKARAN, S. SAHNI, *Randomized routing, selection and sorting on the OTIS-Mesh optoelectronic computer*,IEEE transaction on Parallel and Distributed Systems, Vol. 9, No.9, 1998, pp. 833–840.

[14] W. HENDRICK, O. KIBAR, P. MARCHAND, C. FAN, D. V. BLERKOM, F. MCCORMIC, I. COKGOR, M. HANSEN, S. ESENER, *Modeling and Optimization of the Optical Transpose Interconnection System*,Optoelectronic Technology Center, Program Review, Cornell Univ, 1995.

Edited by: Marcin Paprzycki
Received: Nov 16, 2007
Accepted: Nov 21, 2007

SCALABLE COMPUTING: PRACTICE AND EXPERIENCE
Volume 9, Number 1, pp. 77–82. http://www.scpe.org
ISSN 1895-1767
© 2008 SCPE

AIRBORNE SOFTWARE: COMMUNICATION AND CERTIFICATION

ANDREW J. KORNECKI*

Abstract. This paper discusses the distributed nature of airborne software intensive systems. The concept of fly-by-wire, the issues of safety, component communication, and certification are on the forefront of modern airborne applications. Since the communication between variety of on-board systems and subsystems is of paramount importance, the paper discusses the properties of new bus standard based on the Ethernet protocol gaining recognition and widespread use in large commercial aircraft. Avionics Full Duplex Switched Ethernet (AFDX) addresses quality of service, determinism, and reliability required in an aircraft environment. AFDX has reduced the cost of aircraft data networks, increased network capability, and supported determinism required by the airborne system certification guidelines system.

Key words. avionic bus architecture, airborne software certification, software safety

1. Introduction. When the Wright brothers took the first flight in December 1903, the flying controls were moved by mechanical strings. With progress of aeronautical technologies, the flight-control surfaces have been moved by hydraulic devices fed by hydraulic lines that run through the airplane. A fly-by-wire (FBW) system is a computer-based flight control system that eliminates the mechanical and hydraulic links between the cockpit controls and the airplane moving surfaces. Pulleys, cranks, wires and hydraulic pipes are replaced by much lighter electrical wires. The Lunar Module in 1969 was the first flying machine to use FBW concept. This concept was subsequently applied on the F9 Crusader in 1972 and then successfully implemented in all military aircraft. Obviously, FBW requires communication between distributed aircraft subsystem and components. We discuss the distributed nature of airborne systems, the issues of safety, recent directions in bus communication between airborne subsystems, and certification.

2. Fly-By-Wire. Fly-by-wire defines not only an electrically-signaled control system. Today, FBW identifies computer-configured controls with a computer system processing the pilot's inputs in accordance with software programs and producing outputs controlling the aircraft control surfaces or engine. Computers allows for fast processing compensating for the lack of natural aerodynamic stability e.g. in modern military aircraft. The FBW systems save weight and thus reduce fuel consumption, due to the elimination of the bulk and mechanical complexity of the linkages connecting the pilot's stick/yoke to the control surfaces. It also facilitates multiple aircraft configurations increasing aerodynamic efficiency and providing better performance. However, this may result in the aircraft becoming unstable over part of the range of speed and altitude conditions. FBW systems overcome this by including high-integrity automatic stabilization of the aircraft to compensate for the loss of natural stability and provide the pilot with good control and handling characteristics.

Since commercial introduction of fly-by-wire systems by Airbus 320 and then Boeing 777, demand on data transmission between avionics subsystems has increased exponentially. Modern aircraft include large number of systems consisting of sensors, transducers, actuators, alerts and warnings, electrical and hydraulic power control, information management system, interfaces and other digital electronics. Direct connections with ground and satellite communication, links to convey weather and traffic data place additional burden on the number of systems in modern aircraft. Reliability and timely delivery of those communications are essential to safety of the aircraft operations.

3. Software Safety Issues. Software for high integrity digital FBW systems can account for between 60% and 70% of the total development cost of the complete FBW system. It is due to the size and complexity of the software required to implement the flight control functions and the problems associated with establishing the safety of the software. The software functions may be divided into three basic areas: control laws, built-in-test, and redundancy management. Flight control laws, representing the functional aspect of the system, account for 25% to 30%, while the built-in-test accounts for around 10% of the total software. Thus over 60% the code account for configuration and redundancy management [1]. Some of these tasks involved in failure detection and isolation, and reconfiguration in the event of a failure include: sensor data validation, failure detection and consolidation, sensor failure isolation and system reconfiguration, cross lane data transfer, computer output voting and consolidation, iteration period synchronization, recording of fault data, system status and control.

The FBW system must be no less safe than the mechanical control systems, which it replaces. The statistical level of civil aircraft safety, derived from the total number of civil aircraft crashes occurring in a year from all causes divided by the total number of aircraft flying and their annual operating hours, corresponds to $1 \times 10^{-6}/hour$. The mean time

*Embry Riddle Aeronautical University, Daytona Beach, FL 32114, USA (kornecka@erau.edu).

between failures (MTBF) of a single channel FBW system is about 3,000 hours. The FBW system must therefore incorporate redundancy with multiple parallel channels so that it is able to survive at least two failures. Assuming sufficient redundancy, it may be acceptable to fly with one failed channel.

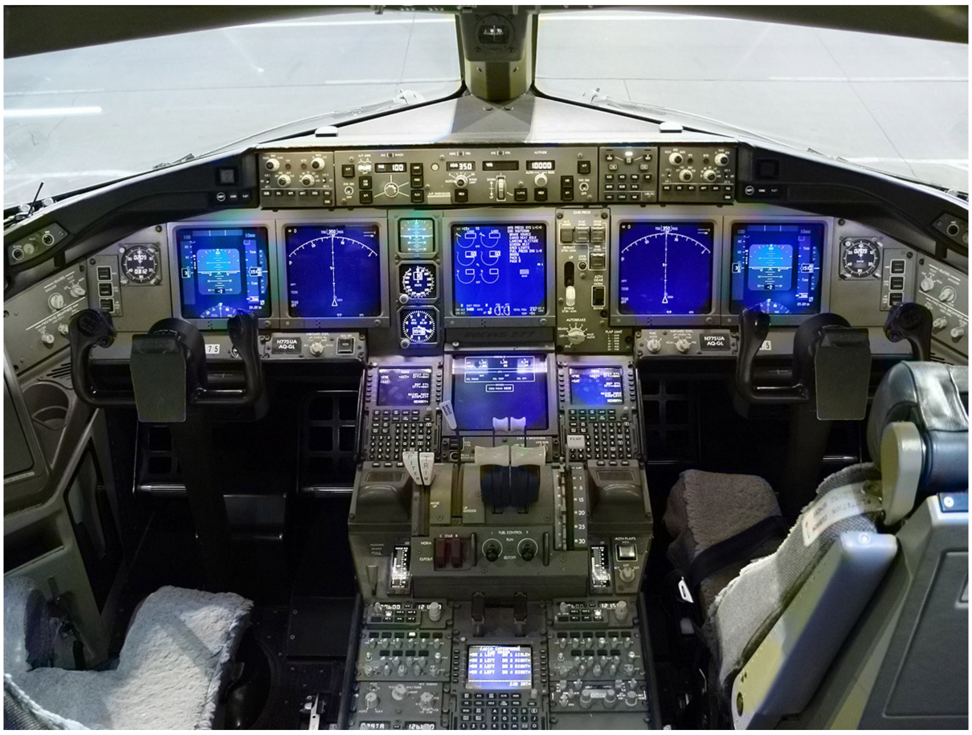


FIG. 3.1. *Cockpit view of modern aircraft (B777)*

4. Communication and Bus Standards. Aeronautical Radio, Inc. (ARINC) was chartered to serve as the airline industry's single licensee and coordinator of radio communication outside of the government. ARINC took on responsibility for all ground-based, aeronautical radio stations and for ensuring station compliance with FRC rules and regulations. Currently, ARINC Standards specify the air transport avionics equipment and systems used by more than 10,000 commercial aircraft worldwide [2].

The flight control, navigation, display computers, and any other equipment on board have to communicate with each other. In early systems, a dedicated wire, i. e. point-to-point connection, was required for each separate component that needed to exchange information with any other. For example, three components exchanging data with each other required a total of six data lines. Bus architecture provided a better solution to integrate the various system components. ARINC 429 defines unidirectional communication bus protocol between a transmitter and up to 20 receivers connected by a two-wire data bus. ARINC 429 has been serving aviation very well and it is used in a majority of commercial aircraft designed before mid-80. However, further advancement of computing capabilities and the need for a multi-transmitter protocol with higher transmission rate to accommodate large number of data in triple-redundant Boeing 777 flight control system led to creation of ARINC 629 based on the Digital Autonomous Terminal Access Communication (DATAC) development work of Boeing and NASA [3]. Each transmitting component's data was coded and broadcast in a synchronized order and each computer or component could be programmed to retrieve only the needed information. The increased data rate and ability to connect more terminals were an added value. However, the proprietary nature of the design hindered widespread use of this standard.

5. Ethernet Solution: AFDX. In the early 90's when Airbus began research for the development of the A380 model, extensive time and resources was put towards implementing new technologies that were reliable and safe without imposing production delays and budget overruns. This resulted in the creation of Avionics Full Duplex Switched Ethernet (AFDX) based on the existing Ethernet technology while requiring less wiring, reducing weight, lowering cost, and increasing communication speeds.

The Ethernet IEEE 802.3 standard with abundance of COTS components, maturity, and the standardization would serve reduction of both cost and development time. However, Ethernet cannot provide aircraft data networks with the required by safety/time critical aviation system "guarantee of service" [4]. Although the data rate is higher than established avionics buses, the potential for transmission collision and related need for re-transmission is significant enough to prevent its defined use in an avionics system. Additionally, a collision could cause time delays exceeding defined time constraints and effectively preventing the transmission from ever reaching its destination—thus "leading to non-deterministic behavior" [5].

Avionics Full Duplex Switched Ethernet (AFDX) extensions to the IEEE Standard 802.3 (Ethernet) address quality of service and reliability required in an aircraft environment. AFDX provides much higher data rates than existing avionics. Both Airbus and Boeing are developing currently aircraft that use AFDX. ARINC 664 Part 7 defined AFDX as a standard.

Each system to be connected to the aircraft bus includes a dedicated End System, which is a gateway to connect each avionic system Line Replaceable Unit (LRU) via the AFDX Switch to the network for the purpose of receiving or transmitting. The End System can accommodate multiple subsystem partitions embedded within each avionics system.

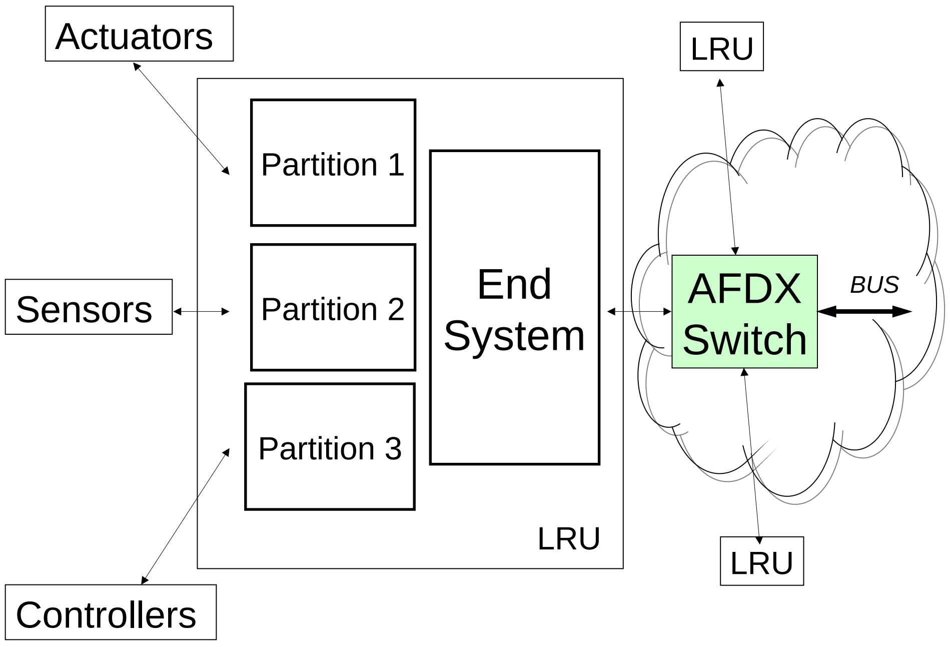


FIG. 5.1. *Avionic Computer System Line Replaceable Unit with three partitions [5]*

To prevent the issues of non-determinism AFDX implements a full-duplex switched Ethernet "... in which the maximum amount of time it would take any one packet to reach its destination is known" [5] thus implementing collision free deterministic system. To achieve this, each subsystem is connected to a switch via two twisted pairs, one dedicated for transmission and one to reception. The AFDX switch implements scheduling protocol controlling the allocation of transmission packets and assuring determinism of the bus operation.

While use of full duplex protocol prevents collisions, it does introduce inaccuracy of packer transmission timing i. e. jitter due to the delay introduced by one packet waiting for another to be transmitted. To maintain system determinism the jitter must be controlled.

AFDX switches play a critical role maintaining and loading configuration tables defining the system and subsystem. Additionally, switches can be cascaded to form larger hierarchical network. The main functions of AFDX switch are:

- Set up a point to point connection network between a sender and one or more receivers
- Establish communication between subscribers on the network
- Monitor and control bandwidth checking and maintaining data frame integrity

Virtual Links (VL) are the channels with unique identifiers which carry messages from source through the switch to the destination. Together with a switch, they effectively create virtual point-to-point network composed of a single source

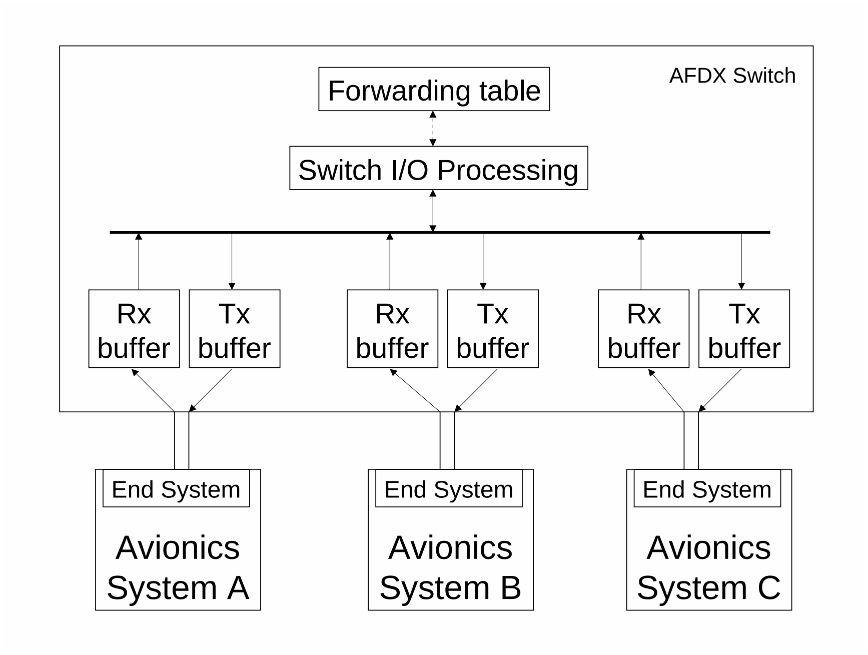


FIG. 5.2. *Full Duplex AFDX Switch Connection with three LRU's [5]*

and delivered to one or more specific receivers. Even though there can only be once source for each VL, it is permissible to originate more than one VL from each source creating a point-to-point connection to one or more destinations. More than one VL can be transmitted at the same time through the physical Ethernet link.

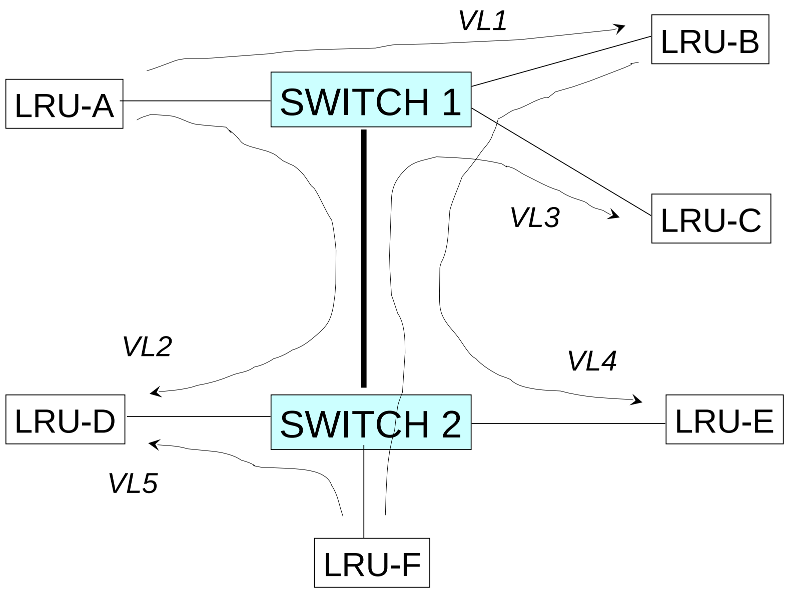


FIG. 5.3. *Virtual Links between six LRU's in a system with two switches*

Each VL has a well defined transmission time, which guarantees [4]:

- Reserved path into the switched network topology
- Reserved bandwidth into the global AFDX network
- Global time scheduling between End Systems

- Known maximum latencies and jitters
- Access delay to the network
- End-to-end delivery, without acknowledgements nor retries

The bandwidth control is achieved by the timing constraints associated to each VL. Every VL is allocated a predefined Bandwidth Allocation Gap (BAG) representing the minimum time interval between the starting bits of two successive AFDX frames (assuming no jitter) and defined in the configuration tables. The BAG values are predetermined for each Virtual Link and the End System allocates the data frames to each BAG to regulate the traffic. The scheduler calculates the maximum jitter for a BAG to accommodate all frames within a given VL. Certainly, to maintain deterministic network, the total bandwidth of all created VL cannot exceed the maximum available bandwidth of the network. When VL is transmitted, AFDX scheduling takes place considering each VL parameters i. e. BAG and the Largest Ethernet Frame. By limiting the rate at which Ethernet frames can be transmitted on a specific VL and by limiting the size of the Ethernet frames, the scheduler can maintain a deterministic network. The scheduler must also coordinate the multiplexing of all transmissions to maintain the total jitter introduced below acceptable levels.

6. Certification. In highly regulated industries such as aerospace, nuclear, medical and transportation it is critical to assure the system safety [6]. In civil aviation industry, a rigorous certification process follows the guidelines of the Radio Telecommunication Committee for Aviation (RTCA) [7]: DO-178B for software and DO254 for hardware. DO-178B was developed by the avionics industry to provide software considerations for aircraft systems including software controlled computing devices. DO-254 provides hardware deployment guidelines for systems including microprocessors, or other complex electronic devices (FPGA, PLA, ASIC, etc.), are deployed in aircraft equipment designs. Both document place significant emphasis on the design/development and verification process to assure product safety and thus constitute guidelines for the design/development assurance. For validation and testing there is an additional need to conform to environmental qualification, as per DO-160D, and need for rigorous and complete testing of variety of failure recovery situations. The system safety considerations, due to lack of well-established metrics, are most difficult to evaluate. Two automotive industry standards [8], SAE ARP 4754 and SAE ARP 4761, give fundamental guidelines to the system safety considerations.

The initial selection of the specific measurable criteria to help in assessment of the data buses is proposed in the CAST paper [9]: safety, data integrity, performance, design/development assurance, and validation/testing approaches. Data integrity and performance can be demonstrated by specific array of tests. In the process, the allowed error rate per byte should be defined and means to recover from the errors should be provided. The load analysis and related bus capacity should be also specified. The extreme cases of bus loss, shortening, and opening should be considered in the analysis and tests. Each specific data bus may have details that need to be addressed by a particular method discussed in advance between the applicant and the appropriate certification authority.

7. Conclusions. Avionics Full-Duplex Switched Ethernet (AFDX) has defined the future of aircraft data networks. Based on a mature technology with extensions to meet fly-by-wire systems' needs, AFDX has reduced the cost of aircraft data networks, increased network capability, and supported determinism required by the airborne system certification guidelines system. AFDX has increased the reliability of avionics systems, while reducing implementation time and cost. The concept and implementation underwent full certification scrutiny and as such has gained stamp of approval for the safety critical applications.

Acknowledgement. The author would like to acknowledge ERAU MSE graduate student Leonard Matos whose term paper was instrumental in collection of the presented material.

REFERENCES

[1] A. J. Kornecki, K. Hall: *Approaches to Assure Safety in Fly-by-wire Systems: Airbus vs. Boeing,* Proceeding of the Eight IASTED International Conference on Software Engineering and Applications (SEA 2004), ISBN: 0-88986-427-6, MIT, Cambridge, MA, November 2004 `http://faculty.erau.edu/korn/papers/SEA04Kornecki_436-021.pdf`

[2] *ARINC Aeronautical Radio Inc.,* `https://www.arinc.com/cf/store/catalog.cfm?prod_group_id=1&category_group_id=3`
[a.] ARINC Specification 629 Part 1–5—Multi-Transmitter Data Bus, Part 1—Technical Description,
[b.] ARINC Specification 664 Part 7, Aircraft Data Network, Avionics Full Duplex Switched Ethernet (AFDX) Network.

[3] D. C. E. Holmes: *Global System Data Bus Using the Digital Autonomous Terminal Access Communication Protocol,* IEEE/AIAA 7th Digital Avionics Systems Conference and Technical Display, Fort Worth, Texas, 13–16 October 1986, pp. 227–233, 1986.

[4] Creative Electronic Systems S. A.: *CES White Paper on ADFX,* 2003, `http://www.ces.ch/documents/downloads/afdx_white_paper.pdf`

[5] Condor Engineering: *ADFX Protocol Tutorial,* Condor Engineering, Inc., 2005, `http://www.acalmicrosystems.co.uk/whitepapers/sbs8.pdf`

[6] A. J. Kornecki, J. Zalewski: *Avionics Databus Safety Criteria and Certification,* Proceedings of ESREL'05 Conference, Advances in Safety and Reliability, Gdańsk, Poland, June 2005, ISBN-0415383404, pp. 1149–1155 http://faculty.erau.edu/korn/papers/ZalewskiKorneckA4.pdf

[7] RTCA Radio Telecommunication Committee for Aviation, Inc., http://www.rtca.org/downloads/ListofAvailableDocs_WEB_OCT%202007.htm

[a.] RTCA/DO-178B. Software Considerations in Airborne Systems and Equipment Certification. Washington, DC, 1992.

[b.] RTCA/DO-254. Design Assurance Guidance for Airborne Electronic Hardware. Washington, DC, 2000.

[c.] RTCA/DO-160D, Environmental Conditions and Test Procedures for Airborne Equipment, Washington, DC, 1997.

[8] SAE Society of Automotive Engineers,

[a.] SAE ARP 4754, Certification Considerations for Highly-Integrated or Complex Aircraft Systems, 1996, http://www.sae.org/technical/standards/ARP4754

[b.] SAE ARP 4761, Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment, 1996, http://www.sae.org/technical/standards/ARP4761

[9] CAST Position Paper CAST-16. Databus Evaluation Criteria, 2003. http://www.faa.gov/certification/aircraft/av-info/software/CAST/cast-16.rtf

Edited by: Marcin Paprzycki
Received: Nov 20, 2007
Accepted: Nov 21, 2007

AIMS AND SCOPE

The area of scalable computing has matured and reached a point where new issues and trends require a professional forum. SCPE will provide this avenue by publishing original refereed papers that address the present as well as the future of parallel and distributed computing. The journal will focus on algorithm development, implementation and execution on real-world parallel architectures, and application of parallel and distributed computing to the solution of real-life problems. Of particular interest are:

Expressiveness:

- high level languages,
- object oriented techniques,
- compiler technology for parallel computing,
- implementation techniques and their efficiency.

System engineering:

- programming environments,
- debugging tools,
- software libraries.

Performance:

- performance measurement: metrics, evaluation, visualization,
- performance improvement: resource allocation and scheduling, I/O, network throughput.

Applications:

- database,
- control systems,
- embedded systems,
- fault tolerance,
- industrial and business,
- real-time,
- scientific computing,
- visualization.

Future:

- limitations of current approaches,
- engineering trends and their consequences,
- novel parallel architectures.

Taking into account the extremely rapid pace of changes in the field SCPE is committed to fast turnaround of papers and a short publication time of accepted papers.

INSTRUCTIONS FOR CONTRIBUTORS

Proposals of Special Issues should be submitted to the editor-in-chief.

The language of the journal is English. SCPE publishes three categories of papers: overview papers, research papers and short communications. Electronic submissions are preferred. Overview papers and short communications should be submitted to the editor-in-chief. Research papers should be submitted to the editor whose research interests match the subject of the paper most closely. The list of editors' research interests can be found at the journal WWW site (`http://www.scpe.org`). Each paper appropriate to the journal will be refereed by a minimum of two referees.

There is no a priori limit on the length of overview papers. Research papers should be limited to approximately 20 pages, while short communications should not exceed 5 pages. A 50–100 word abstract should be included.

Upon acceptance the authors will be asked to transfer copyright of the article to the publisher. The authors will be required to prepare the text in $\LaTeX 2_\varepsilon$ using the journal document class file (based on the SIAM's `siamltex.clo` document class, available at the journal WWW site). Figures must be prepared in encapsulated PostScript and appropriately incorporated into the text. The bibliography should be formatted using the SIAM convention. Detailed instructions for the Authors are available on the SCPE WWW site at `http://www.scpe.org`.

Contributions are accepted for review on the understanding that the same work has not been published and that it is not being considered for publication elsewhere. Technical reports can be submitted. Substantially revised versions of papers published in not easily accessible conference proceedings can also be submitted. The editor-in-chief should be notified at the time of submission and the author is responsible for obtaining the necessary copyright releases for all copyrighted material.