Biblioteka Informatyki
Szkół Wyższych

Information Systems Architecture and Technology

Intelligent Information Systems, Knowledge Discovery, Big Data and High Performance Computing

Library of Informatics of University Level Schools

Series of editions under the auspices
of the Ministry of Science and Higher Education

The ISAT series is devoted to the publication of original research books in the areas of contemporary computer and management sciences. Its aim is to show research progress and efficiently disseminate current results in these fields in a commonly edited printed form. The topical scope of ISAT spans the wide spectrum of informatics and management systems problems from fundamental theoretical topics to the fresh and new coming issues and applications introducing future research and development challenges.

The Library is a sequel to the series of books including Multidisciplinary Digital Systems, Techniques and Methods of Distributed Data Processing, as well as Problems of Designing, Implementation and Exploitation of Data Bases from 1986 to 1990.

Wrocław University of Technology

Information Systems Architecture and Technology

Intelligent Information Systems, Knowledge Discovery, Big Data and High Performance Computing

Editors
Leszek Borzemski
Adam Grzech
Jerzy Świątek
Zofia Wilimowska

Wrocław 2013

Publication partly supported by

Faculty of Computer Science and Management
Wrocław University of Technology

Project editor

Arkadiusz GÓRSKI

The book has been printed in the camera ready form

OFICYNA WYDAWNICZA POLITECHNIKI WROCŁAWSKIEJ
Wybrzeże Wyspiańskiego 27, 50-370 Wrocław
http://www.oficyna.pwr.wroc.pl;
e-mail: oficwyd@pwr.wroc.pl
zamawianie.ksiazek@pwr.wroc.pl

ISBN 978-83-7493-800-6

CONTENTS

INTRODUCTION

Intelligent Information Systems, Knowledge Discovery, Big Data and High Performance Computing are key challenges in today's IT research and development. New IT developments and design paradigms in these domains are very crucial in business because they provide pioneering ways to boost business performance to be more competitive on the market.

The advance of Intelligent Information Systems (IIS) runs especially in the Internet based environments where new concepts, models, services and application are developing every day. In this book we present issues related to different aspects of IISs, including characterization of the new architectures, use of artificial intelligence in tackling challenging problems, knowledge discovery and data mining as well as the Big Data and high performance computing issues.

This book consists of chapters presenting a balanced coverage of emerging IIS problems concerning:

Part 1. *Intelligent Information Systems and Knowledge Discovery*
Part 2. *Artificial Intelligence and Multiagent Systems*
Part 3. *Big Data and High Performance Computing*

PART 1: INTELLIGENT INFORMATION SYSTEMS AND KNOWLEDGE DISCOVERY

Chapter 1 presents a concept of using knowledge models to secure intellectual capital in enterprise based on Decisional DNA that will allow the company to accumulate knowledge and protect its' previously gained intellectual capital. Decisional DNA is a domain-independent and flexible knowledge representation structure. Its main features of acquiring and storing experiential knowledge of formal decisional events are used to deal with unexpected situations, especially, convert unstructured data into well-structured knowledge, including, information from websites. This work focuses on tools based on SOEKS (Set of Experience Knowledge Structure) and Decisional DNA that will allow the company to accumulate knowledge and protect its' previously gained intellectual capital. They will be also designed to enable delegating duties to less experienced employees by providing them with an instrument that will allow them to independently perform more complicated tasks.

Chapter 2 describes a user-centric approach to the extraction of modal equivalences as linguistic summarizations of huge amounts of data. The user orientation is realized by use of semi-natural language statements. The approach follows from an original proposal formulated in a series of previous works in which modal literals, modal conjunctions, modal inclusive and exclusive alternatives as well as modal conditionals were used to capture the result of knowledge extraction from relational data repositories. The originality of this approach follows from the fact that the existing data resources to be summarized are treated as empirical knowledge bases developed by autonomous systems (regardless from their actual nature) and pragmatically interpreted as dialogue systems. These dialogue systems are assumed to be equipped with semi-natural language processing modules producing narrow class of semi-natural language statements being commonsense interpretations of the above mentioned modal formulas.

Chapter 3 presents an approach to integrating fuzzy structures. There is introduced a multi-agent system with observer agents and one main agent. Observer agents inform main agent about observed objects using linguistic values. Main agent's task is to integrate gathered data and propose a few mutually exclusive solutions if data is inconsistent. Two strategies for validating data consistency based on Silhouette and GAP Statistics are proposed. Integration process is performed using consensus based method. A prototype platform was implemented and obtained results are presented.

Chapter 4 demonstrates an approach to the problem of integration of knowledge expressed in form of modal formulas with operators for possibility, belief and certainty. The authors assume that an agent is collecting messages describing properties of an external object from a group of other agents in order to create his own prepositional attitude. The language of communication allows sentences in form OP(p), where OP is a modal operator of possibility, belief or certainty, and p is logical conjunction, alternative or exclusive alternative of two prepositional variables. The receiving agent collects the knowledge using a form of internal representation. A simple consensus method is applied in order to integrate the gathered knowledge, and the resulting consensus is subsequently used to create a set of sentences that can be communicated by the agent, according to the rules described in works concerning the problem of grounding modalities. A simple framework for testing this algorithm was developed. Sample results of integration are presented.

Chapter 5 deals with the multi-agent framework for inconsistent sensor data integration. Data is integrating into a consistent set of easily accessible and manageable records. The main process is based on consensus algorithm which converts data using summation minimization. Principal agent not only integrates data, but also checks whether the level of consistency in the data set is acceptable or not and may propose alternative solutions with higher level of consistency. To achieve this task agent uses clustering methods. A prototype platform performing described integration was implemented and some achieved results are presented.

Chapter 6 proposes and evaluates new algorithms to generate network graphs with two predefined properties, degree distribution and motifs distribution. The results may be used to fine-tune simulations modeling epidemic spread on networks, information cascades and other dynamic network phenomena.

PART 2: ARTIFICIAL INTELLIGENCE AND MULTIAGENT SYSTEMS

Chapter 7 introduces a new approach for evaluating similarity (or distance) between documents described with rough sets. The task focuses on an interpretation of the border of a rough set and its influence on common knowledge processing tasks. This problem can be met in automated distributed knowledge processing. Then to reduce processing time a descriptive complexity of documents can be reduced using approximate descriptions by means rough sets.

Chapter 8 deals with a multi-agent system involved in a collective alignment of naming conventions. Highly distributed and autonomous systems need to be endowed with the ability to establish and share utilized language structures. In particular, following a general model of the naming convention alignment process (language game model) we overview the behavior of the multi-agent system. The major focus of this research is on the problem of how to describe an individual linguistic stance of an agent (agent-level) and relate it to a collective stance (system-level). Thus allowing for a quantitative research of the alignment process, both on agent- and system-level. In particular, the author introduces and briefly discusses six distinct measures: success rate, language coherence rate, average number of used words, overall number of words, amount of synonymy and homonyms, strength of language associations and language strength.

Chapter 9 provides a brief overview of selected structures aimed at knowledge representation in the form of ontologies based on description logic and aims at comparing them with their counterparts based on the rule-based approach. The formalisms of the OWL language were used to record ontologies, while the rules were expressed in Prolog. To better illustrate these two ways of knowledge representation, examples of best practices from the field of IT services management were used, which are contained in a set of publications known as the Information Technology Infrastructure Library (ITIL). The purpose of the comparison was to examine the possibility of using an ontological approach in situations where the use of rule-based solutions is problematic.

Chapter 10 presents α-ISIS – a ground-up reimplementation of the micro-CDS-ISSI database system. The CDS-ISIS database is a popular software system, used for generalized information storage and retrieval. It has been maintained by UNESCO since 1985. It was designed mainly for bibliographical applications. At the time of its implementation its notable features were advanced text searching capabilities, complex thesauri, and multi-linguality. The objective of this work was to open the system to new technologies (including novel storage databases, UNICODE standard, XML data representation), and make α-ISIS suitable for large databases. A set of new features has been added on the functional level, mainly – remote document indexing, multi-database search, dynamic meta-indexes, web services support and cloud support via MongoDB.

Chapter 11 gives an outline of a multiagent platform for knowledge integration. The input knowledge is obtained from a distributed group of agents in a form of semi-hierarchical partitions. Information about the world is gathered by observer-agents and later sent to central agent to form a knowledge profile. The central agent integrates the knowledge profile using methods adapted from consensus theory in order to choose a fitting representative. Central agent deals with inconsistency of information. First, the agent rates a consistency of the knowledge profile and if the consistency level is not satisfying, a clustering process initiates. Once the knowledge profile becomes divided into clusters, a separate representative is evaluated for each of its consistent parts. A method for choosing optimal number of clusters is presented. A prototype of the platform has been developed in order to provide empirical results.

PART 3: BIG DATA AND HIGH PERFORMANCE COMPUTING

Chapter 12 deals with a database management problem of handling large amounts of data while providing short response time. Problem is not only proper manner of storing records but also efficient way of processing them. In the meantime GPUs developed computational power many times greater than that offered by comparable CPUs. In this research the authors investigated benefits that using GPU in database queries execution can give. Using offered in PostgreSQL database User-Defined Aggregate extension mechanism, they implemented own versions of 3 standard aggregates: sum, average and population standard deviation that were executed on GPU by using OpenCL. They found that, while in simple aggregates (sum and average) there was no gain, in more complex aggregates (population standard deviation) they were able to achieve more than 2 times shorter execution time than in standard build in database aggregate.

Chapter 13 presents an approach to computerized detection and correction of errors in Polish texts. Most contemporary editors provide solutions that prove ineffective if misspelled word transforms into another, valid word. Resulting sentence may be grammatically or semantically incorrect, even though all words are correct. This phenomenon is especially common in Polish due to rich inflection, complex case system and the use of diacritical marks. Error detection and disambiguation among correction candidates may require context-dependent analysis, based on syntax or even semantics. To introduce effective methods of error detection, typical causes of errors and possible error patterns are considered. Proposed method of error detection and correction is based on suitably adapted Dependency Grammar. It allows to capture syntactic and, partially, semantic dependencies in sentence, even between distant words or phrases. Therefore, inflectional properties of linked words may be analyzed and best word selected form the list of correction candidates. Presented approach may be used as supplement tool in automated text processing large-scale systems.

Chapter 14 describes a framework that utilizes Python and new extensions for JavaScript to delegate code to be executed by a web site clients and gathers results on a server. It means that it provides a fully functional basic environment for employing parallel distributed computing in the WWW environment. Using this technology to get access to the incredible computing power that lays in the web client devices connected to the computer network that utilize just only a part of their actual power.

Chapter 15 shows results of research on the problem of time-optimal tasks scheduling and resources allocation in parallel machines system. A parallel machine system consisting of m parallel machines is considered. This system can execute n tasks. All *n* tasks are independent and the number of tasks is greater than number of machines. It is also assumed that is constancy of resources allocation in execution time all tasks set. For some tasks processing time function the mathematical model of this problem is formulated. Because the problem belongs to the class of NP-complete problems an heuristic algorithm for solution this problem is proposed. Some results of executed numerical experiments for basis of proposed heuristic algorithm are presented

Chapter 16 discusses the problem of load balancing in distributed data warehouse systems. Original load balancing algorithms are presented: the Adaptive Load Balancing Algorithms for Queries (ALBQ) and the algorithm that uses grammars and learning machines in managing the ETL process. These two algorithms build the load balancing based on queries analysis, however the methods of query analysis are quite different. While ALBQ is based on calculation of computing power and available system assets, the gaSQL algorithm includes direct grammar analysis of the SQL query language and its classification using machine learning. The WINE-HYBRIS algorithm that uses the CUDA architecture and Cloud Computing will be presented as a platform for developing the gaSQL algorithm.

Chapter 17 presents ASvis system – a web-based large graph search engine and visualizer. The motivation to construct search engine and visualizer using new Web technologies were connectivity data between existing autonomous systems (AS Autonomous System) on the Internet, collected by the Division of Distributed Computer Systems, Institute of Informatics, Wroclaw University of Technology. The application to search and visualize connections between ASes (ASvis) consists of two parts: the backend – an application running on the server, and the frontend – application running in a web browser. The backend is responsible for providing the data to frontend. Communication frontend to backend is possible by interface using software architecture standard REST (Representational State Transfer). ASvis application shows that the new web technologies enable advanced programming tasks done that previously were the domain of desktop applications.

Chapter 18 shows the performance of computing cluster, consisting of computers connected with network, using graphics cards to perform computations. Preparation of research environment included development of cluster, installation of environments (MPI, CUDA) and the implementation of *k*-means algorithms in versions: sequential, and parallelized (on GPU, MPI and mixture of MPI with GPU). Research consisted of measuring the execution time of the algorithms and calculation of metrics to evaluate the performance: efficiency and speedup. An influence of the data transfer on effectiveness of the algorithm parallelized on a cluster, was examined.

This book contains the contributions accepted after the review of authors' submissions. We hope that the book will be considered as a forum for presentation of original and professional work in emerging research areas such as Intelligent Information Systems, Knowledge Discovery, Big Data and High Performance Computing and many others creating innovative advances in commerce, science, and society.

We would like to express many thanks to reviewers who helped to evaluate the submissions.

We thank all the authors who have submitted their contributions to be published in this book.

Wrocław, September 2013

Leszek Borzemski

PART 1

INTELLIGENT INFORMATION SYSTEMS AND KNOWLEDGE DISCOVERY

Decisional DNA,
Set of Experience Knowledge Structure

Norbert BOLEŃSKI*
Edward SZCZERBICKI**

A CONCEPT OF USING KNOWLEDGE MODELS TO SECURE INTELLECTUAL CAPITAL IN ENTERPRISE

Without a doubt, knowledge has become the most important resource for all companies in the world. Knowledge extended of our own observations and practice is being called experience. With the increase of knowledge and gained experience, the individual becomes more and more valuable employee in terms of business. The above mentioned knowledge can be gained only by self-improvement, which requires time and other necessary elements for the implementation of the process of cognition. Possibility of acquiring new knowledge by practice is strongly related with the working area environment. Employers may create the proper environment within their companies and provide employees with necessary tools, yet it generates high cost. However, technological development which has took place in recent years along with legislation changes being consequences of globalization, resulted in an increase of mobility of knowledge, experience and individuals who possess both. At times when global labor market is open for highly skilled employees, the employer is not always able to keep the desired, experienced employee at his company. Therefore, it is highly important from the company's point of view to be capable to secure intellectual capital prior to its loss with an eventual loss of the employee.

The paper is an initial concept of planned doctorate research, focused on feasibility of development of tools based on SOEKS (Set of Experience Knowledge Structure) and Decisional DNA that will allow the company to accumulate knowledge and protect its' previously gained intellectual capital. Those tools will be also designed to enable delegating duties to less experienced employees by providing them with An instrument that will allow them to independently perform more complicated tasks. The tool that planned for this research is called by the Authors a Virtual Mentor (VM).

1. BACKGROUND AND THE NEED FOR THE CONCEPT OF VIRTUAL MENTOR MODEL: THE CASE OF PRODUCER AND SUPPLIER

We live in times of common and rapid increase of knowledge and information acquisition, rapid development of the various fields of management, and where techno-

* Gdansk University of Technology, Gdansk, Poland.
** The University of Newcastle, Newcastle, Australia.

logical progress is growing exponentially. This determines the extensive growth of competitiveness among markets and companies. Number of entrepreneurs, focusing on their own "modern" image, place emphasis on innovation and investing in the newest technologies, patents, etc. However, the implementation of new technologies and tools leads not only to the improvement of enterprise's image, but it may influence the quality of company's management as well. This might be especially important for those companies, where the choice among various contractors is determined by factors other than financial. In many aspects of business management, the transfer of knowledge and experience between the companies can be run almost in one-to-one relation. Nevertheless, there are some areas of business activity (especially highly specialized), where, due to the large number of variables coming from the company's internal and external environment, company faces a lot of barriers and unknown factors. In effect, despite the experience previously gained by an individual, the knowledge can be limited to indicating the direction in which one needs to look for effective solution. To take-up production of certain product, the producer develops specifications. In order to meet the requirements of the final customer, the specification described needs to be perfectly completed. To minimize their own costs, the majority of the producers outsource manufacturing intermediate products. Outsourced compounds need to meet defined parameters to obtain final product up to standards described in specification. Following the security policy, producers tend to order different components and intermediate products from different suppliers. Suppliers, on the other hand, produce intermediate products with chemicals or other intermediate products ordered and bought from another companies. Intermediate products are characterized by certain parameters, including classes of purity, which, although they are within the accepted standards, in combination with other compounds, shall require further, specific treatment. Therefore, the same intermediate products may differ from one another, being still within the defined standards. In a result, they may require different procedures to further treatment to obtain desired final effects. As a result, individual lots of intermediate products or compounds need being tested for specific, required parameters.

After receiving raw materials/compounds that will be subsequently used for the production of intermediate products, suppliers/producers need to analyse them in terms of their compliance with the specifications of the final intermediate product. The planned research would focus on methods of further processing, i.e., what should be done to receive the results of the desired range. Due to the number of parameters that need to be taken into account (i.e. processing time, the proportions of combined compounds and reagents, etc.) the required tests are time consuming and involve the use of specialized equipment. Moreover, developing successful methodology for a single sample does not guarantee that it will be equally suitable for the different lot of samples and that it will allow to receive the results on the same level. This comes from the fact, that seemingly the same compounds may differ within the acceptable standards and due to their specific characteristics and properties, may need to be treated differently in further process in order to lead to the same final results. This forces laboratory employees to modify methods and procedures each time, which, in turn, results in additional time consuming effort.

In the process of examining samples, the crucial part is played by the experience of people conducting research in laboratories. In the case of testing a number of compounds, the minimal change of parameters of one of them makes previously developed methodology ineffective. However, the more experienced the laboratory personnel is, the less time is required to succeed in modifying method formulated before or develop new methods. This is especially true, if the laboratory employees have already examined compounds or intermediate products from the given supplier. Therefore, it is highly important for less experienced employees that they are able to benefit from the knowledge of their colleagues. However, the process of mutual exchange of experiences can be long-lasting and, in many cases, is determined by personality traits and mentor-student type of relation. It is believed that in the times when the advance in technology enables knowledge formalization, is it possible to create a virtual model of a mentor/virtual library of information for the business sector, in which the experience of the employees is a key element in building a competitive advantage. One of the major advantages of such solution, apart from the quick and limitless access to the available resources of knowledge within the company, is the ability to save the knowledge of all employees – regardless of their occupation or the time, when they have been employed. All employees would have their contribution to the development of this model, especially in the initial, developing phase of research that would include "brainstorm", giving the opportunity to all employees to demonstrate their experience and knowledge. The model allowing also unwanted results to be saved, would not only be a certain "library" of previous research, but it would also allow the laboratory personnel to analyse the impact of various factors on another.

There are many different sources of knowledge in enterprises. According to Richard F. Bonner and Andrew J. Baborski we can distinguish knowledge acquired from [1]:

- Employees;
- Technology – information about the possibilities and requirements for goods, knowledge is divided into branches having regard to the division on raw materials, machinery, planning;
- Environment – knowledge of market trends, suppliers, competitors and consumers or recipients;
- Structures – natural order of performing some actions/procedures;
- Law;
- Regulations;
- Other.

This paper focuses on possibilities of securing knowledge (which is not only "the feature" of an individual, but it can be successfully ascribed to the organization) gained from the first three of these above presented sources. In order to succeed in recording and securing intellectual capital of the company, the need of using knowledge model, including SOEKS (Set of Experience Knowledge Structure) and DDNA (Decisional DNA) [2, 3] is suggested

The organization of the paper content is as follows: in the next section the concept of Set of Experience and Decisional DNA presented by Szczerbicki and Sanin is explained. The third section deals with the concept of Virtual Mentor. The paper is finished with brief conclusions.

2. SOEKS AND DDNA

Using the rules for the transcription of the genetic code (DNA) in natural world Szczerbicki and Sanin [4, 5] conceived the concept of decision-making DNA by implementing the principles of DNA's encoding, storage and inference. Architecture also provides the ability to share knowledge between units.

The key component of DDNA is known as SOEKS (Set of Experience Knowledge Structure), the set of combinations of variables, functions, constraints and rules (Fig. 1).

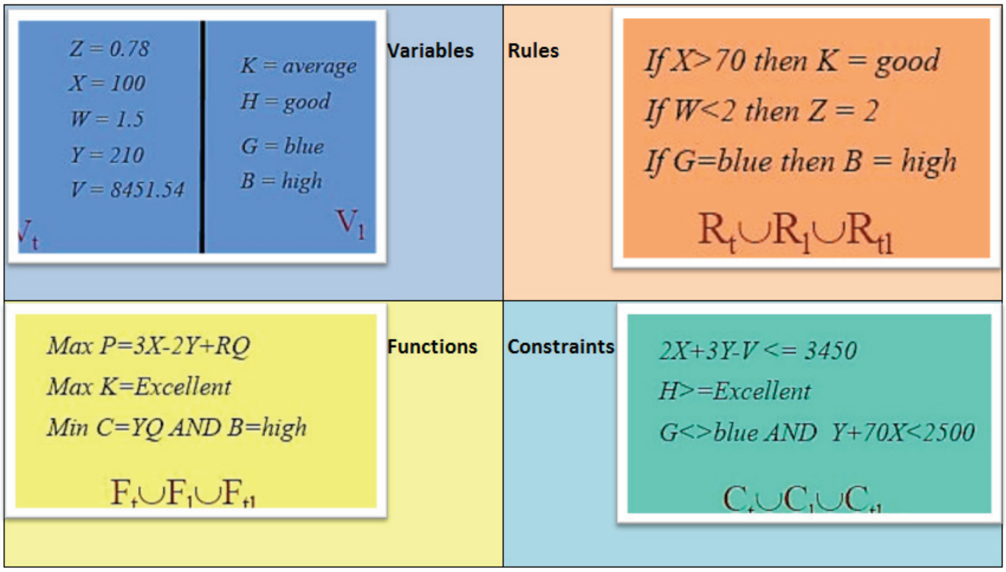


Fig. 1. Components of experience representation [4, 5]

It has been shown that Knowledge Representation (KR) is a medium of translation between human expression and computers [6]. It relies on determining results, instead of actions. It can facilitate the process of making decisions and recommendations. KR is a set of ontological commitments to answer questions of how to interpret the real world. SOEKS has been designed in accordance with the fundamentals of the concept of KR [5, 6]. It is a flexible and independent knowledge representation which can handle information and knowledge of differing format, held within organisations/companies. SOEKS is intended to collect experiences and knowledge from multiple applications that are assembled as formal decision events, in order to assist organizations to make precise decisions, predictions, and recommendations. As it is commonly known, human DNA carries genetic information within the combination of its four elements. SOEKS uses an analogy to natural DNA to extract knowledge and arrange it in combinations of four elements: variables, functions, rules, and constraints. It is an eminently suitable tool for knowledge management tasks. Moreover, it has also been used to collect and store formal decisional events in an explicit manner [7]. Its ontology can be expressed in XML (Extensible Markup Language) or OWL (Ontology Web Language), in order to make it shareable and transportable [7]. SOEKS is defined by the structure as shown in Fig. 2:

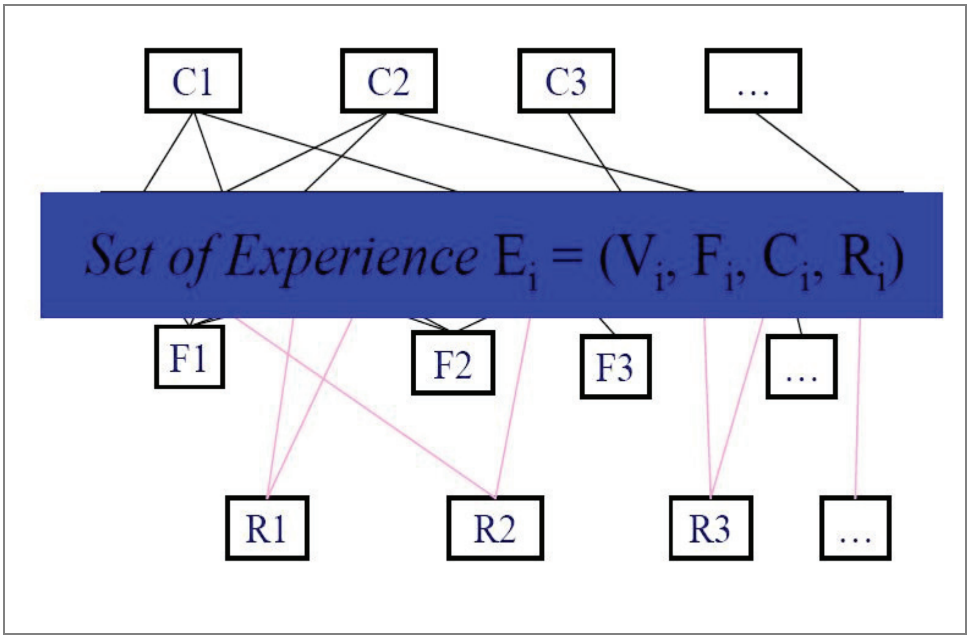


Fig. 2. Set of Experience [4, 5]

Functions are made up of interactions between variables, which include dependent variables and a set of input variables. Constraints are another way of representing associations between variables. Although a constraint is some form of a function, it has a different purpose. It limits the performance and configuration of a system and restricts the feasible solutions in a decision problem. Lastly, rules are another way to express links between variables. They condition possible relationships that operate on the universe of variables. In other words, they use the statements IF-THEN-ELSE to connect conditions with their consequences.

Additionally, the structure of SOEKS is analogous to some important features of natural DNA. It imitates a gene in combining four nucleotides of DNA by integrating four components of experience to adapt to different needs. The components are not isolated, but there are connections between them. In the same way as a gene produces a phenotype, set of experience yields a value of decision with its elements. Each SOEKS can be categorised and stores data as would a gene in DNA [4, 5]. A set of experiences in the same category makes up a decisional chromosome, which stores decisional strategies for that category. Each module of chromosomes establishes an entire inference tool to offer a blueprint of knowledge inside an organisation [5, 6].

A similarity metric is one of the fundamental concepts of Knowledge Discovery (KD). It provides a way to improve the effectiveness and efficiency of organisational strategies and operations. Common similarity methods employ geometrical functions to measure mathematical distance between a pair of objects and to find an optimal object for the prediction or decision. SOEKS introduces this approach to calculate individual similarities between variables, functions, constraints and rules, and to produce a joint similarity value. This provides a scalar measure of the similarity between two objects which is ultimately used for prediction purposes in data mining processes [6].

Decisional DNA secures the intellectual capital within the company by:

- Conserving the historical knowledge
- Preserving the decision-making experience,
- Insights for future users to clear stock of knowledge through Knowledge Representative Software.

3. THE CONCEPT OF VIRTUAL MENTOR (VM): INFERENCE MODEL

The company's experience is being accumulated over the years (Fig. 3), while the company develops – by acquiring skills, building relationships with partners, developing habits. Interacting with the environment (customers, suppliers, etc.), thereby enriching the company with new experiences, is a natural consequence of business activity. Therefore, it can be assumed that the greater is the number of interactions with the environment, the greater experience of the company. The same situation can be applied to particular employee, who over the time, becomes more and more experienced and therefore, more and more valuable for his/her employer.

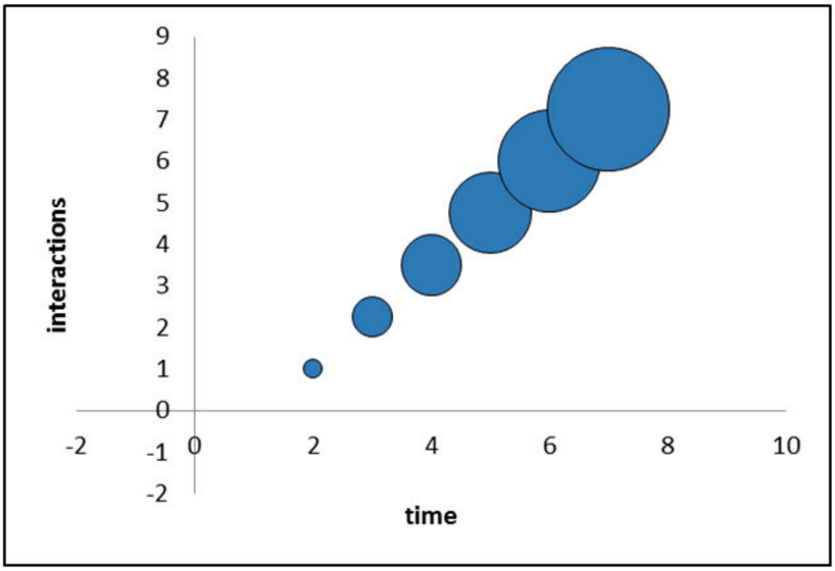


Fig. 3. Experience level in time

Virtual Mentor is planned as a specific tool that can be used to collect the experience acquired by all employees of the company. It can be recognised as a company's most valuable capital as well as a library of knowledge for its' employees. The model powered by data, processes them and presents the best solution based on past experience. Employees can benefit from its' resources, supplying it with new data after the use. This results in a self-calibration, as it is highly unlikely to receive two identical orders. Thus, it can be assumed that each next result, will be better and will bring company closer to the desired result. It should be noted that the learning model is based on empirical research and gradual, consistent entering new data.

The differences between individual orders relates to two areas – information received from the client, who defines the desired parameters of the final product, and the information from the suppliers, who provide intermediate products, compounds and raw materials of certain parameters. Information from the customers should be filtered in order to eliminate information irrelevant from the company's point of view. Information from suppliers should be analysed in order to choose the most accurate supplier. Filtering and analysing the information from the environment, each time brings the company closer to obtaining a desired result.

The below given diagram presents the path of given signal (final product) and highlights the way the raw materials and intermediate products of various parameters cover before becoming part of final product. It can be assumed that each lot is different because [Fig. 4]:

a) it may come from different source to a higher level (different pathway of intermediate products),
b) variations in quality and parameters of intermediate products across one source – may have different parameters, within the margin of error.

As each set of Experience is a set of elements with different characteristics, each new experience should be individually analysed and recorded.

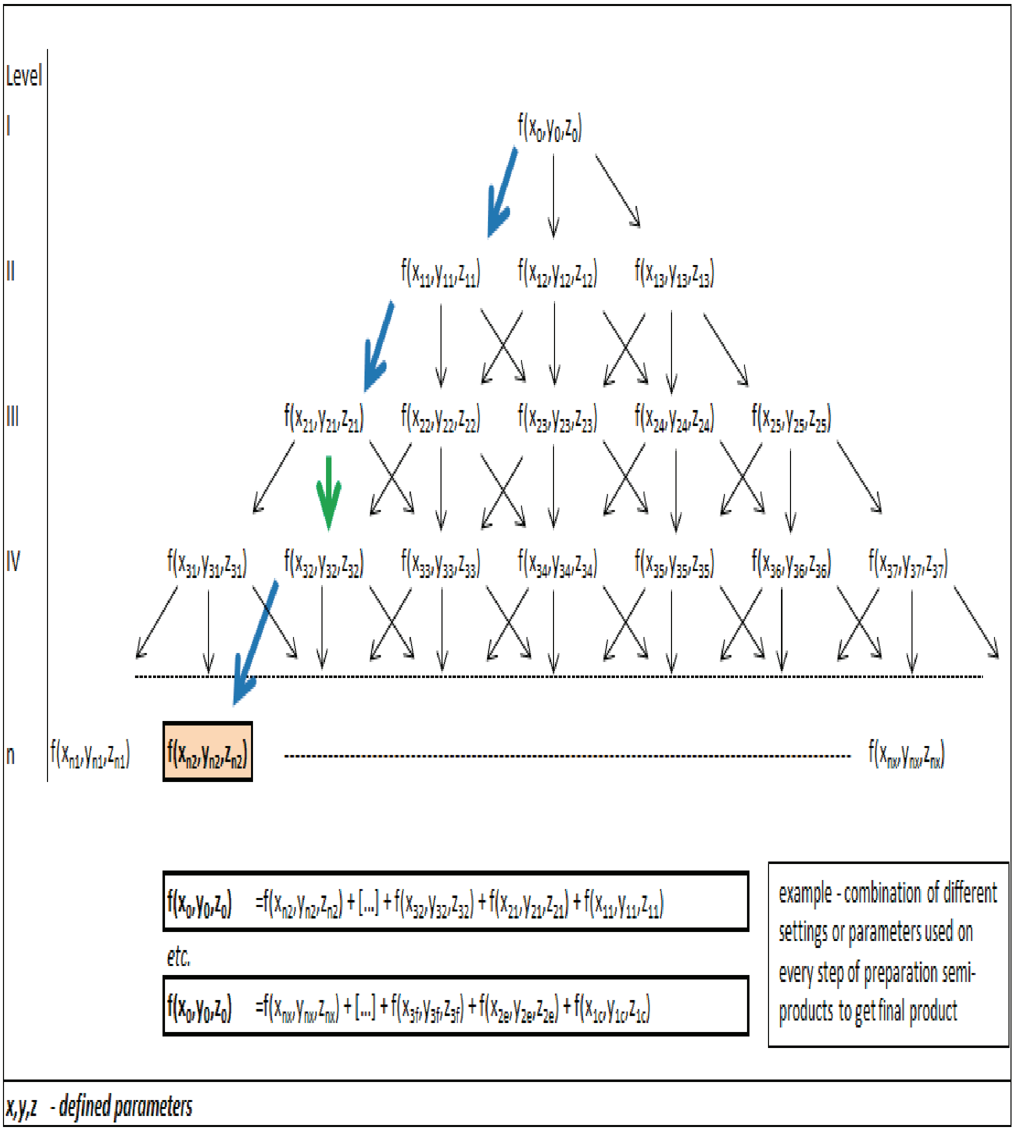

Level
I
II
III
IV
n
$f(x_0,y_0,z_0)$
$f(x_{11},y_{11},z_{11})$ $f(x_{12},y_{12},z_{12})$ $f(x_{13},y_{13},z_{13})$
$f(x_{21},y_{21},z_{21})$ $f(x_{22},y_{22},z_{22})$ $f(x_{23},y_{23},z_{23})$ $f(x_{24},y_{24},z_{24})$ $f(x_{25},y_{25},z_{25})$
$f(x_{31},y_{31},z_{31})$ $f(x_{32},y_{32},z_{32})$ $f(x_{33},y_{33},z_{33})$ $f(x_{34},y_{34},z_{34})$ $f(x_{35},y_{35},z_{35})$ $f(x_{36},y_{36},z_{36})$ $f(x_{37},y_{37},z_{37})$
$f(x_{n1},y_{n1},z_{n1})$ $f(x_{n2},y_{n2},z_{n2})$ $f(x_{nx},y_{nx},z_{nx})$
$f(x_0,y_0,z_0) = f(x_{n2},y_{n2},z_{n2}) + [...] + f(x_{32},y_{32},z_{32}) + f(x_{21},y_{21},z_{21}) + f(x_{11},y_{11},z_{11})$
etc.
$f(x_0,y_0,z_0) = f(x_{nx},y_{nx},z_{nx}) + [...] + f(x_{3f},y_{3f},z_{3f}) + f(x_{2e},y_{2e},z_{2e}) + f(x_{1c},y_{1c},z_{1c})$
example - combination of different settings or parameters used on every step of preparation semi-products to get final product
x,y,z - defined parameters

Fig. 4. Function of Input and Output Values

Knowledge Representation requires creating appropriate interface which will define both, the output values (parameters to be obtained) and input (parameters of intermediate products) (Fig. 5).

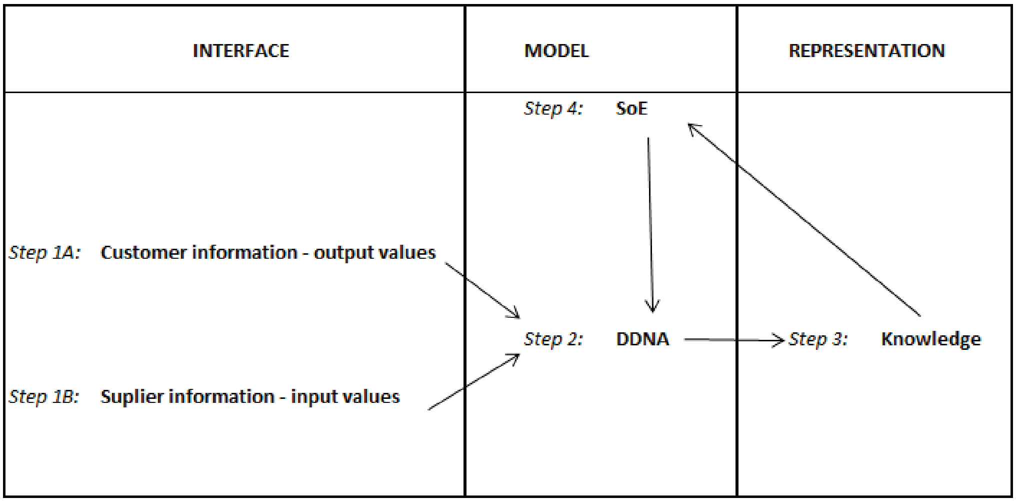

INTERFACE
MODEL
REPRESENTATION
Step 4: SoE
Step 1A: Customer information - output values
Step 2: DDNA
Step 3: Knowledge
Step 1B: Suplier information - input values

Fig. 5. The way of learning model

Model presented in Fig. 5 performs the inference process, presenting the most relevant results, which after verification will create another Set of Experience.

Huge amount of data and information spread by different sources force todays enterprises to create effective tools for representation of knowledge (KR). Knowledge Representation, due to the possibilities offered by intelligent reasoning, is the most effective method of improving knowledge management. The proposed concept of VM combined with SOEKSA and DDNA carries a promise to enhance the process of KM very significantly.

4. CONCLUSION

It is believed that we possess technology and the necessary tools to develop detailed model that may protect the company's intellectual capital from escaping and at the same time improve the access to our past experience. The proposed VM is such a model. When fully developed it will help meeting the needs of the client through processing and combining the available data about the needs of customers with data about the elements that we have from suppliers to achieve the goal of minimizing the time and costs involved in the process.

REFERENCES

[1] ABRAMOWICZ W., NOWICKI A., OWOC M., *Zarządzanie wiedzą w systemach informacyj-nych*, Wydawnictwo Akademii Ekonomiczne im. Oskara Langego we Wrocławiu, Wrocław 2004, pp. 21–23.

[2] SANIN C., SZCZERBICKI E., TORO C., *An OWL Ontology of Set of Experience Knowledge Struc-ture*, Journal of Universal Computer Science, Vol. 13, No. 2, 2007, pp. 214–216.

[3] SANIN C., SZCZERBICKI E., *Experience-based Knowledge Representation: SOEKS*, Cybernetics and Systems: An International Journal, 2009, **40**(2), pp. 99–122.

[4] SANIN C., SZCZERBICKI E., *Application of a Multi-domain Knowledge Structure: The Decisional DNA*, [in:] N. Nguyen, E. Szczerbicki (eds.), *Intelligent Systems for Knowledge Management*, Springer Berlin–Heidelberg, 2009, pp. 65–86.

[5] SANIN C., SZCZERBICKI E., *Decisional DNA and the Smart Knowledge Management System: A process of transforming information into knowledge*, [in:] A. Gunasekaran (ed.), *Techniques and Tools for the Design and Implementation of Enterprise Information Systems*, 2008, pp. 149–175.

[6] WANG P., SANIN S., SZCZERBICKI E., *Introducing The Concept of Decisional DNA-Based Web Content Mining*, Cybernetics and Systems: An International Journal, 43, 2012, pp. 97–117.

[7] DUONG T.H., NGUYEN N.T., JO G., *Constructing and mining a semantic-based academic social network*, Journal of Intelligent & Fuzzy Systems, 21(3), 2010, pp. 197–207.

linguistic summarization, modal language
modal equivalence, language grounding

Dominik WIĘCEK
Radosław Piotr KATARZYNIAK*

MODAL EQUIVALENCES AS LINGUISTIC SUMMARISATION OF DATA RESOURCES

In this paper we describe a user-centric approach to the extraction of modal equivalences as linguistic summarizations of huge amounts of data. The user orientation is realized by use of semi-natural language statements. Our approach follows from an original proposal formulated in a series of previous works in which modal literals, modal conjunctions, modal inclusive and exclusive alternatives as well as modal conditionals were used to capture the result of knowledge extraction from relational data repositories. The originality of our approach follows from the fact that the existing data resources to be summarized are treated as empirical knowledge bases developed by autonomous systems (regardless from their actual nature) and pragmatically interpreted as dialogue systems. These dialogue systems are assumed to be equipped with semi-natural language processing modules producing narrow class of semi-natural language statements being commonsense interpretations of the above mentioned modal formulas.

1. INTRODUCTION

In one of our previous papers [7] an original model for modal conditional extraction from huge amounts of data was presented. The target of that paper was to provide a theoretical model for an effective application of interpreted modal conditionals as linguistic summarizations of huge repositories of relational data. The model presented in [7] went beyond the classic case of conditional (rule) extraction in the following sense: At first, instead of the single rule extraction “if *p* then *q*”, all semantically correlated modal conditionals were taken into account simultaneously to capture the complete set of extracted relations {“if *p* then *q*”, “if *p* then not-*q*”, “if not-*p* then *q*”, and “if not-*p* then not-*q*”}. At second, the result of each association rule extraction was assumed to be communicated to end user as a natural language

* Institute of Informatics, Wrocław University of Technology, Skwer Idaszewskiego 1, 50-370 Wrocław.

statement rather than as a pair of two formal symbols p and q with a related numerical measure of its vagueness (precision and/or completeness). At third, the resulting sets of linguistic representations had to fulfil some intuitive consistency requirements known from the discourse of natural language. At fourth, some access restrictions to overall collection of relational items in data bases were assumed, reflecting practical and technical constraints known from actual circumstances. The first three assumptions correspond to end-user centric perspective in knowledge systems design where the role of natural language communication between end user and system would be strongly recommended. The fourth assumption reflected (at the theoretical level) restrictions of multiple natures (e.g. technical and pragmatic). *In this paper we apply similar approach to another case of knowledge representation structures, namely modal equivalences.*

Let us assume that a database $D = \{d_1, d_2, ..., d_n\}$ is given. Let the target of extraction be finding all pairs of statements p and q such that p holds for a member of D if and only if q holds for the same member of D. In classic approach we would expect one of the following two alternative results from such extraction:

- The systems informs that $p \Leftrightarrow q$ holds for each item in D;
- The systems informs that $p \Leftrightarrow q$ does not hold for at least one item in D.

It is quite clear that to yield this result (in particular the first case of linguistic summarization) the knowledge extraction processes need to process the complete data set D. The question arises what happens when the complete processing of data set D is not possible due to some constraints in data access? A deeper discussion can prove that in order to capture results of partial knowledge extraction one would probably use the following modal extensions of equivalence $p \Leftrightarrow q$:

Case 1. (For all what we already got to know) it is possible that p holds if and only if q holds: $Pos(p \Leftrightarrow q)$.

Case 2. (For all what we already got to know) we believe that p holds if and only if q holds: $Bel(p \Leftrightarrow q)$.

What remains is:

Case 3. (For all what we already got to know) we know that p holds if and only if q holds: $Know(p \Leftrightarrow q)$.

Obviously, Case 3 covers the situation in which complete processing of data set D is possible and actually realized.

Below a brief presentation of a model is given in which all three cases of modal equivalence extraction are captured in an effective and consistent way. The core element of this model follow the results given in [2]–[4].

2. EMPIRICAL KNOWLEDGE BASE AND STATE OF PROCESSING

It is assumed in our model that possible states of an object $d \in D$ can be described by a vector of properties $P_1, ..., P_K$. The empirical knowledge about d is collected over time and can be given as the following temporal collection [5]:

Definition 1. The overall *state of empirical knowledge* about the object collected by the knowledge system up to the time point t is represented by the following temporal collection:

$$KS(t) = \{BP(t_n): t_n \in T \text{ and } t_n \leq t\}. \tag{1}$$

where

$$BP(t) = <P_1(t), P_2(t), ..., P_K(t)>. \tag{2}$$

is a relational structure that represents a piece of knowledge about the object. Each piece of knowledge is related to a particular time point t and called *t-related base profile*. The following interpretations and constraints have apply to *t-related base-profiles*:

- For each $i = 1, ..., K$, both $P_i(t) \in \{1, 0, \varepsilon\}$ and $P_i(t) \in \{1, 0, \varepsilon\}$ hold.
- For each $i = 1, ..., K$, $P_i(t) = 1$ holds if and only if the knowledge system got to know that the object exhibited property P_i at the time point t.
- For each $i = 1, ..., K$, $P_i(t) = 0$ holds if and only if the knowledge system got to know that the object did not exhibit property P_i at the time point t.
- For each $i = 1, ..., K$, $P_i(t) = \varepsilon$ holds if and only if the knowledge system had got no knowledge about state of property P_i at the time point t.

This definition relates directly to another definitions for base profiles used elsewhere e.g. [2], [3], [5], [6]. Moreover, following these works, we assume that at each state of the knowledge system's life cycle (in a particular time points $t \in T$) the empirical knowledge base is always divided into shallow (probably already processed) and deep (still unprocessed) body of knowledge. This partition is crucial for practical reasons and seems to be strongly supported by multiple cognitive theories of mind e.g. see verbal-nonverbal distinction [1], deep-shallow conceptual structures [8], etc. We capture this concept by the following definition:

Definition 2. Let $KP(t)$ be called the *knowledge processing state* (KP-state). At each particular time point $t \in \mathrm{T}$, the related KP-state is given as a partition of $KS(t)$ into the shallow and deep subarea of knowledge, represented by the following tuple:

$$KP(t) = (\overline{KP}(t), \underline{KP}(t)) \,. \tag{3}$$

where $KS(t)$ states for the experience which is located at the shallow cognitive level (at the time point *t*) and *KS*(*t*) states for the remaining experience (located at the deep cognitive level and at the same *t*). An obvious consequence is that the following two equations hold:

$$\overline{KP}(t) \cup \underline{KP(t)} = KS(t)\,, \tag{4}$$

$$\overline{KP}(t) \cap \underline{KP(t)} = \varnothing. \tag{5}$$

In our multiple works (e.g. [2], [3], [5]) we strongly stressed that the whole *KS*(*t*) should constitute the actual knowledge body from which all language summarizations related to object *d* are extracted. *Unfortunately, KP-state is also strongly involved in this extraction and plays substantial role in the final choice of related modal markers.* Detailed definitions that shape the way in which extraction of modal equivalence $Pos(p \Leftrightarrow q)$, $Bel(p \Leftrightarrow q)$ and $Know(p \Leftrightarrow q)$ should be technically realized are given in forthcoming sections. Similar solutions may be proposed for remaining cases of modal equivalences e.g. $Pos(p \Leftrightarrow \neg q)$ or $Know(\neg p \Leftrightarrow q)$.

3. THE GROUNDING SETS AS RELEVANT DATA FOR PROPER EXTRACTION

It follows from the theory of modal languages grounding [4] that each modal equivalence should be extracted from a particular and relevant body of adequate knowledge from *KS*(t). For instance, all modal extensions of the equivalence $p \Leftrightarrow q$ should be extracted from empirical data (a base profiles collection) in which the conjunctive condition $P(t) = 1$ and $Q(t) = 1$ or $P(t) = 0$ and $Q(t) = 0$ is fulfilled. None of these conditions falsifies the equivalence. Obviously, the strength with which this data supports the equivalence $p \Leftrightarrow q$ increases whenever the overall number of these base profiles increases. However, similarly to other classes of formulas the final choice of operator from the set {*Know*, *Bel*, *Pos*} depends on relations between the grounding sets of all four semantically related equivalences $p \Leftrightarrow q$, $p \Leftrightarrow \neg q$, $\neg p \Leftrightarrow q$, and $\neg p \Leftrightarrow \neg q$. This idea should be applied to extract modal linguistic summarizations, too. To capture this fact we need the concept of the so-called grounding sets: (see also [9]).

Definition 3. Let *p* and *q* be symbols of language interpreted as linguistic names for properties *P*, $Q \in \{P_1, P_2, ..., P_K\}$, $P \neq Q$, respectively. Let the following sets be given:

$$C = \{BP(\hat{t}) \in KS(t) : \hat{t} \boxed{7} t \wedge P(\hat{t}) \in \{1, 0\} \cup Q(\hat{t}) \in \{1, 0\}\} \tag{6}$$

$$\overline{C} = C \cap \overline{KP} \tag{7}$$

$$\underline{C} = C \cap \underline{KP} \tag{8}$$

Definition 4. Let *p* and *q* be symbols of language interpreted as linguistic names for properties *P*, *Q* ∈ {P_1, P_2, ..., P_K}, $P \neq Q$, respectively. Let the following sets be given:

$$\begin{aligned}
\overline{C_{p\wedge q}} &= \{BP(\hat{t}) \in \overline{C} : P = 1 \wedge Q = 1\} \\
\overline{C_{p\wedge \neg q}} &= \{BP(\hat{t}) \in \overline{C} : P = 1 \wedge Q = 0\} \\
\overline{C_{\neg p\wedge q}} &= \{BP(\hat{t}) \in \overline{C} : P = 0 \wedge Q = 1\} \\
\overline{C_{\neg p\wedge \neg q}} &= \{BP(\hat{t}) \in \overline{C} : P = 0 \wedge Q = 0\}
\end{aligned} \tag{9}$$

$$\begin{aligned}
\underline{C_{p\wedge q}} &= \{BP(\hat{t}) \in \underline{C} : P = 1 \wedge Q = 1\} \\
\underline{C_{p\wedge \neg q}} &= \{BP(\hat{t}) \in \underline{C} : P = 1 \wedge Q = 0\} \\
\underline{C_{\neg p\wedge q}} &= \{BP(\hat{t}) \in \underline{C} : P = 0 \wedge Q = 1\} \\
\underline{C_{\neg p\wedge \neg q}} &= \{BP(\hat{t}) \in \underline{C} : P = 0 \wedge Q = 0\}
\end{aligned} \tag{10}$$

$$\begin{aligned}
C_{p\wedge q} &= \overline{C_{p\wedge q}} \cup \underline{C_{p\wedge q}} \\
C_{p\wedge \neg q} &= \overline{C_{p\wedge \neg q}} \cup \underline{C_{p\wedge \neg q}} \\
C_{\neg p\wedge q} &= \overline{C_{\neg p\wedge q}} \cup \underline{C_{\neg p\wedge q}} \\
C_{\neg p\wedge \neg q} &= \overline{C_{\neg p\wedge \neg q}} \cup \underline{C_{\neg p\wedge \neg q}}
\end{aligned} \tag{11}$$

These sets are called the *(t-related) grounding* sets for conjunctions [2][3]. These sets are used to define the equivalence grounding sets which relative grounding strength is given as follows [4]:

Definition 5. The grounding relative strengths for equivalences *p* ⇔ *q*, ¬*p* ⇔ *q*, ¬*p* ⇔ *q* and ¬*p* ⇔ ¬*q* are given as follows:

$$\lambda^{p\Leftrightarrow q} = \frac{(\ |\overline{C_{p\wedge q}}|\ )}{(\ |C_{p\wedge q} \cup C_{p\wedge \neg q} \cup C_{\neg p\wedge q} \cup C_{\neg p\wedge \neg q}|\ )} \tag{12}$$

$$\lambda^{p\Leftrightarrow \neg q} = \frac{(\ |\overline{C_{p\wedge \neg q}}|\ )}{(\ |C_{p\wedge q} \cup C_{p\wedge \neg q} \cup C_{\neg p\wedge q} \cup C_{\neg p\wedge \neg q}|\ )} \tag{13}$$

$$\lambda^{\neg p\Leftrightarrow q} = \frac{(\ |\overline{C_{\neg p\wedge q}}|\ )}{(\ |C_{p\wedge q} \cup C_{p\wedge \neg q} \cup C_{\neg p\wedge q} \cup C_{\neg p\wedge \neg q}|\ )} \tag{14}$$

$$\lambda^{\neg p \Leftrightarrow \neg q} = \frac{(\ |\overline{C_{\neg p \wedge \neg q}}|\ )}{(\ |C_{p \wedge q} \cup C_{p \wedge \neg q} \cup C_{\neg p \wedge q} \cup C_{\neg p \wedge \neg q}|\ )} \tag{15}$$

3. CONDITIONS OF PROPER LINGUISTIC SUMMARIZATION

The situations in which particular modal equivalences can be used as adequate linguistic summarization of data sets are defined by means of the so-called epistemic satisfaction relation:

Definition 6. Epistemic satisfaction relation $KP(t) \vDash Pos(p(q))$ holds iff:

$$\overline{C_{p \wedge \neg q}} \cup \overline{C_{\neg p \wedge q}} = \varnothing \tag{16}$$

$$\overline{C_{p \wedge q}} \neq \varnothing \tag{17}$$

$$\lambda^{\Leftrightarrow}_{\min Pos} < \lambda^{p \Leftrightarrow q} \leq \lambda^{\Leftrightarrow}_{\max Pos} \tag{18}$$

Epistemic satisfaction relation $KP(t) \vDash Bel(p(q))$ holds iff:

$$\overline{C_{p \wedge \neg q}} \cup \overline{C_{\neg p \wedge q}} = \varnothing \tag{19}$$

$$\overline{C_{p \wedge q}} \neq \varnothing \tag{20}$$

$$\lambda^{\Leftrightarrow}_{\min Bel} < \lambda^{p \Leftrightarrow q} \leq \lambda^{\Leftrightarrow}_{\max Bel} \tag{21}$$

Epistemic satisfaction relation $KP(t) \vDash Know(p(q))$ holds iff:

$$\overline{C_{p \wedge \neg q}} \cup \overline{C_{\neg p \wedge q}} = \varnothing \tag{22}$$

$$\overline{C_{p \wedge q}} \neq \varnothing \tag{23}$$

$$\lambda^{p \Leftrightarrow q} \simeq 1 \tag{24}$$

$$\underline{C_{p \wedge q}} \simeq \varnothing \tag{25}$$

where four numeric values $\lambda_{\min Pos}$, $\lambda_{\max\mathrm{Pos}}$, $\lambda_{\min\mathrm{Bel}}$, $\lambda_{\max\mathrm{Bel}}$ are called equivalence grounding thresholds fulfilling the following *general* requirement:

$$0 < \lambda^{\Leftrightarrow}_{\min Pos} < \lambda^{\Leftrightarrow}_{\max Pos} \leq \lambda^{\Leftrightarrow}_{\min Bel} < \lambda^{\Leftrightarrow}_{\max Bel} \leq 1 \tag{26}$$

Similar definitions hold for the other cases of modal equivalences.

It has also be strongly stressed that the original theory of grounding defines in a very strict manner some additional requirements for equivalence modality thresholds to ensure desirable properties of linguistic summarizations production. This issue is not discussed in this paper (see [2], [3] for other examples).

3. REMARK ON COMPUTING APPROXIMATIONS OF UNPROCESSED CONTENT

The major problem with an effective application of the above approach to modal equivalence extraction follows from the nature of unprocessed level of existing empirical knowledge: if all empirical knowledge has been processed in order to extract a particular equivalence, then any application of modal operators *Bel* and *Pos* is not possible. The knowledge of the relation between *P* and *Q* is complete and only *Know* operator should be used. At the same time operators *Bel* and *Pos* can be used, if the access to some levels of empirical data is constrain, but at the same time this directly inaccessible data influences indirectly the choice between *Bel* and *Pos*. *At this point the task of modal equivalences extraction refers us to some advanced modern theories and models of natural language production* (e.g. [10]).

From the technical point of view multiple strategies can be design to deal with effective evaluation of $\underline{C_i}$. Below three rather naive approaches are *cited* to illustrate in a very simple way the nature of this task. These strategies were originally presented in [7]

Definition 7 (shallow level projection). Let $\overline{\omega_i}$ be defined by the formula

$$\overline{\omega_i} = \frac{card(\overline{C_i})}{card(\overline{KP})} \tag{27}$$

The shallow level projection strategy assumes that the cardinality of $\underline{C_i}$ can be estimated by means of the approximation formula:

$$card(\overline{C_i}) \simeq \overline{\omega_i} * card(\underline{KP}) \tag{28}$$

This strategy assumes that the distribution of deep-level grounding knowledge is the same as this one located at the shallow level.

Definition 8 (deep level sampling strategy). Let $\alpha \geq 1$ denote the so-called sampling step. Then the number of possible samples Γ to be taken from $\underline{KP}$ is determined by the following formula:

$$\Gamma = [card(\underline{KP}) / \alpha] \tag{29}$$

Let the following sample sets of base profiles from KP be considered:

$$\Phi^{p\wedge q} = \bigcup_{i=0}^{\Gamma-1} \{BP(\alpha i) = 1 \wedge Q(\alpha i) = 1\}\,, \tag{30}$$

$$\Phi^{p\wedge\neg q} = \bigcup_{i=0}^{\Gamma-1} \{BP(\alpha i) : P(\alpha i) = 0 \wedge Q(\alpha i) = 1\}\,, \tag{31}$$

$$\Phi^{\neg p\wedge q} = \bigcup_{i=1}^{\Gamma-1} \{BP(\alpha i) : P(\alpha i) = 0 \wedge Q(\alpha i) = 1\} \tag{32}$$

$$\Phi^{\neg p\wedge\neg q} = \bigcup_{i=0}^{\Gamma-1} \{BP(\alpha i) : P(\alpha i) = 0 \wedge Q(\alpha i) = 0\}\,, \tag{33}$$

provided that the base profiles from $\underline{KP}$ are additionally indexed by numbers from 0 to $card(\underline{KP}) - 1$. The deep level sampling strategy assumes that the cardinality of $\underline{C_i}$ can be estimated by means of the following formula:

$$card(\underline{C_i}) \simeq \alpha * card(\Phi^j). \tag{34}$$

The next example strategy proposed in [7] combines in a certain way the shallow level projection with the deep level sampling into one estimation of inaccessible empirical content.

Definition 9 (shallow-level projection with deep sampling). Let $\overline{\omega_i}$ and χ be defined by the formulas:

$$\overline{\omega_i} \frac{card(\overline{C_i})}{card(\overline{KP})} * 100\% \tag{35}$$

$$\frac{card(\overline{KP})}{card(KP)} \tag{36}$$

and $\underline{\omega_i}$ be given as in Def.10. The shallow level projection strategy with deep level sampling assumes that the cardinality of $\underline{C_i}$ can be estimated by means of the approximation formula:

$$card(\underline{C_i}) \simeq (\chi * \overline{\omega_i} + (1-\chi) * \underline{\omega_1}) * card(\underline{KP}). \tag{37}$$

Obviously multiple and more advanced strategies for estimation of deep content influence on shallow content one can and should be designed. In particular, these strategies may utilize artificial neural networks to reflect in technical systems at least some of the natural phenomena known from modern theories of natural language production and comprehension [10].

3. FINAL REMARKS

In this paper an original user-centric approach to the extraction of modal equivalences was presented. The extracted modal equivalences are treated as linguistic summarizations of huge amounts of data. The user orientation was realized by use of semi-natural language statements communicating the intuitive meaning of extracted formulas.

This approach follows from an original proposal formulated by one of the authors in another series of previous works in which modal literals, modal conjunctions, modal inclusive and exclusive alternatives, and modal conditionals were used to capture the result of knowledge extraction from relational data repositories. The originality of this approach follows from the fact that the existing data resources to be summarized are empirical knowledge bases developed by autonomous systems. These systems are treated as dialogue systems and are assumed to be equipped with semi-natural language processing modules producing certain class of semi-natural language statements being commonsense interpretations of various formulas.

In this paper a particular attention was paid to the problem of utilization of this part of an original theory of grounding by which the influence of deep knowledge processing on shallow knowledge processing was modelled. Brief and intentionally simplified examples of possible implementation techniques were cited to illustrate the way in which consistent sets of modal equivalences can be produced as external knowledge structures. Some references to similar works by one of the authors were given.

ACKNOWLEDGEMENTS

This paper was supported by Grant no. N N519 444939 funded by Polish Ministry of Science and Higher Education (2010-2013).

REFERENCES

[1] FREEMAN W.J., *A neurobiological interpretation of semiotics: meaning, representation, and information*, Information Sciences, Vol. 124, No. 1–4, 2000, 93–102.

[2] KATARZYNIAK R., *On some properties of grounding uniform sets of modal conjunctions*, Journal of Intelligent & Fuzzy Systems, Vol. 17, No. 3, 2006, 209–218.

[3] KATARZYNIAK R., *On some properties of grounding non-uniform sets of modal conjunctions*, Int. Journal of Applied Mathematics and Computer Science, Vol. 16, No. 3, 2006, 399–412.

[4] KATARZYNIAK R., *The language grounding problem and its relation to the internal structure of cognitive agents*, Journal of Universal Computer Science, Vol. 11, No. 2, 2005, 357–374.

[5] KATARZYNIAK R., NGUYEN N.T., *Reconciling inconsistent profiles of agent's knowledge states in distributed multiagent systems using consensus methods*, Systems Science, Vol. 26, No. 4, 2000, 93–119.

[6] KATARZYNIAK R., PIECZYŃSKA-KUCHTIAK A., *Grounding and extracting modal responses in cognitive agents: 'and' query and states of incomplete knowledge*, International Journal of Applied Mathematics and Computer Science, Vol. 14, No. 2, 2004, 249–263.

[7] KATARZYNIAK R., WIĘCEK D., *An Approach to Extraction of Linguistic Recommendation Rules – Application of Modal Conditionals Grounding*, Lecture Notes on Artificial Intelligence, Vol. 7653, 2012, 249–258.

[8] PAIVIO A., *Mental representations: a dual coding approach*, Oxford University Press, New York, 1986.

[9] SKORUPA G., KATARZYNIAK R., *Applying possibility and belief operators to conditional statements*, Lecture Notes on Artificial Intelligence, Vol. 6276, 2010, 271–280.

[10] STACHOWIAK F., *Semantic memory and episodic memory by reference to the ontological grounding of the Old and New meta-informative status in the MIC theory*, [in:] *New Standards for Language Studies*, The 3rd MIC Sorbonne workshop (Paris, Nov. 15–16, 2012).

*inconsistency, integration,
consensus, clustering, fuzzy,
multi-agent*

Grzegorz SKORUPA, Radosław KATARZYNIAK,
Łukasz MODLIŃSKI, Mariusz MULKA*

MULTI-AGENT PLATFORM FOR FUZZY STRUCTURES INTEGRATION TASK

We demonstrate an approach to integrating fuzzy structures. We introduce a multi-agent system with observer agents and one main agent. Observer agents inform main agent about observed objects using linguistic values. Main agent's task is to integrate gathered data. She proposes a few mutually exclusive solutions if data is inconsistent. Inconsistency requires data clustering. We propose two strategies for validating data consistency based on Silhouette and GAP Statistics. Integration process is performed using consensus based method. We implemented prototype platform and present obtained results.

1. INTRODUCTION

The task of gathering and analyzing data in a multi-agent system is a difficult and complicated process. The moment of analyzing the gathered data comes with many problems. Inconsistent observations cause difficulties in data analysis. It is because agents providing the observations are autonomous entities and often use heuristic methods. The problem can be solved during the process of integration conducted on the data. Result of integration process, widely known as consensus, is the best representation of gathered data. Main agent of the multi-agent system gathers answers and conduct the process of integration, measuring the consistency beforehand. If the observations are highly inconsistent, there is an option to cluster them. We present a prototype of a platform that realizes that task and shows results. Example of usage of such platform is a survey among workers of a company or institution. The task is to get an overview of the company. Such overview takes into account each of workers'

* Institute of Informatics, Wrocław University of Technology, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław.

opinion on every feature of their work. Analysis may lead to improvements or changes the company shall conduct. We implemented two differently based data integration algorithms for the purpose of comparing them.

Paragraph two presents the problem of data integration task. Paragraph three provides different consensus functions on which our platform works. Paragraph four organizes consistency measuring methods and clustering. Paragraph five presents platform system and its basic functions. Paragraph six shows example results and comparisons of two different clustering methods. Last paragraph summarises the presented material and demonstrates future usages of the platform.

2. KNOWLEDGE INTEGRATION

Main agent conducts the process of knowledge analysis and gathering. The inconsistencies are often approached during that process. Because of them, she cannot determine the state of the analysed part of world. Inconsistency may be solved using data integration. Main agent uses consensus functions to obtain a result.

Main agent works on linguistic variables which are defined and widely discussed in [8]. Observer agents report their answers in a form of vectors of linguistic values. Main agent translates the data she received. For that task she uses some translating functions that are based on her intuition and experience. Translated observations have a form of vectors of brackets. She organizes gathered vectors in a form of matrix where the rows may represent observations and the columns features. Such generated matrix is an essential tool for our integration process.

2.1. CONSENSUS

Searching for consensus is to find the best representation of data set provided by agents. It is to describe the whole set with single consistent representation. Because of the way of obtaining data, it is evident that the data may be inconsistent. Consensus may be perceived as a minimum of some multi-criteria function. Definition of consensus was widely discussed in [4]. As the author defined, consensus is: "Let us assume that there is an universe of objects and its multi-subset. The aim is to find an object, that belongs to the universe, which represents the best mentioned subset."*

Consensus function must satisfy a few general conditions. It is the minimum of all possible representations of the provided space, according to some criterion. Consensus thus reflects in the best way the data it refers to. For our multi-agent system, we analysed three different algorithms for determining consensus and chose one which suits

* Own translation.

our data structure the best. The definition of each algorithm can be found in [5]. We discuss them shortly. The idea of the first algorithm (pages 123–124) is to determine bracket which is minimal cover for all the brackets. Afterwards we search for the subset that minimizes the sum of distances between the data. The second algorithm uses the distance function to minimize the sum from selected brackets to all other brackets. We search for bracket minimizing each of those sums. Such generated bracket will satisfy our consensus conditions. The idea of third algorithm is much similar to idea of the first one. We chose the second algorithm because of its applicability to our knowledge structure.

2.2. INCONSISTENCY OF KNOWLEDGE

Gathered data may contain inconsistencies. Such problem originates from the cognitive nature of agents providing observations. It may lead to uninteresting and poor results which is because the consensus is based on average value. To solve the problem of inconsistency, we shall appeal to human behavior in such situations. Human nature copes with inconsistency in a few different ways. In particular the approach is to remove observations that decrease the knowledge consistency. It is also possible to cluster the input data into separate clusters and then conduct the process of finding consensus on each of separated clusters. We simulated the second approach within cognitive agent. If a few clusters are derived, main agent proposes consensus result for each of the emerged clusters.

3. FINDING CONSENSUS

3.1. KNOWLEDGE STRUCTURE DEFINITION

Linguistic variable is defined by the quintuple (γ, $T(\gamma)$, U, G, M) in which γ denotes the name of the variable; $T(\gamma)$ is the collection of values; U is the universe of the variable; G is a syntactic rule which generates the terms in $T(\gamma)$; M is a semantic rule which assigns each of the linguistic value that is in $T(\gamma)$ its meaning. Let us define a vector of linguistic values $\overline{O_i} = (o_{i,1}, o_{i,2}, \ldots, o_{i,P})$, $o_{i,j} \in T(\gamma_j)$, $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, P$, consisting of P features obtained from N different, cognitive agents. Suppose that the main agent has built-in expert knowledge in form of a compatibility function $c^i_{(\gamma_k,l)}(x) : U_{\gamma_k} \to [0;1]$, for each of linguistic value that is in $T(\gamma_k)$, where $k = 1, 2, \ldots, P$, $l \in T(\gamma_k)$. Let λ_b, $b = 1, 2, \ldots, P$, be a real value that $\lambda_b \in [0;1]$ for all b. And let $z_{(\gamma_k,l)}$ be the set of $x \in U_{\gamma_k}$ that satisfies $c^i_{(\gamma_k,l)}(x) \geq \lambda_k$. We assume that $z_{(\gamma_k,l)}$ is a bracket defined as follows $[z_{(\gamma_k,l)*}, z^*_{(\gamma_k,l)}]$. Now each of $\overline{O_i}$ is defined by

$\overline{v}_i = (z^i_{(\gamma_1, o_{i,1})}, z^i_{(\gamma_2, o_{i,2})}, \ldots, z^i_{(\gamma_P, o_{i,P})})$. Let us define now the matrix $\{\overline{v_i}\}$, $i = 1, 2, \ldots, N$, where the rows represent N observations and the columns P features.

Let C be consensus defined as follows: consensus is the best representation of given set. The wider definition of consensus can be found in [4]. Let $(C_1, C_2, \ldots, C_P)$ be a consensus vector, that C_i, $i = 1, 2, \ldots, P$ is a consensus for each of features, $C_i = \left[z^i_{(\gamma_k, o_w)*}, z^{i\;*}_{(\gamma_k, o_w)} \right]$.

3.2. DATA INTEGRATION PROCESS

We need few more tools to integrate the gathered data. Let us define a distance function which is as follows

$$g(z^i_{(\gamma_k, o_{i,k})}, z^j_{(\gamma_k, o_{j,k})}) = \left| z^{i\;*}_{(\gamma_k, o_{i,k})} - z^{j\;*}_{(\gamma_k, o_{kj,})} \right| + \left| z^i_{(\gamma_k, o_{i,k})*} - z^j_{(\gamma_k, o_{j,k})*} \right| \tag{1}$$

which is a sum of subtractions between upper and lower bracket values. Let us also define a distance function between two vectors of brackets

$$d(v_i, v_j) = \left| g(z^i_{(\gamma_1, o_{i,1})}, z^j_{(\gamma_1, o_{j,1})}) \right| + \left| g(z^i_{(\gamma_2, o_{i,2})}, z^j_{(\gamma_2, o_{j,2})}) \right| + \ldots + \left| g(z^i_{(\gamma_k, o_{i,k})}, z^j_{(\gamma_k, o_{j,k})}) \right| \tag{2}$$

which is a sum of distances between each of argument of vectors.

Definition of consensus function that we use is widely explained in [5] under section 5.4.1 subsection 5.2. The major idea of that algorithm is to minimize

$$\sum_{z^l_{(\gamma_k, o_{l,k})}, l=1,\ldots,N} g(z^i_{(\gamma_k, o_{i,k})}, z^l_{(\gamma_k, o_{l,k})}) =$$

$$= \sum_{z^l_{(\gamma_k, o_{l,k})}, l=1,\ldots,N} \left(\left| z^{i\;*}_{(\gamma_k, o_{i,k})} - z^{l\;*}_{(\gamma_k, o_{l,k})} \right| + \left| z^i_{(\gamma_k, o_{i,k})*} - z^l_{(\gamma_k, o_{l,k})*} \right| \right) = \tag{3}$$

$$= \sum_{z^l_{(\gamma_k, o_{l,k})}, l=1,\ldots,N} \left| z^{i\;*}_{(\gamma_k, o_{i,k})} - z^{l\;*}_{(\gamma_k, o_{l,k})} \right| + \sum_{z^l_{(\gamma_k, o_{l,k})}, l=1,\ldots,N} \left| z^i_{(\gamma_k, o_{i,k})*} - z^l_{(\gamma_k, o_{l,k})*} \right|$$

What means minimizing each of sums $\sum_{z^l_{(\gamma_k, o_{l,k})}, l=1,\ldots,N} \left| z^{i\;*}_{(\gamma_k, o_{i,k})} - z^{l\;*}_{(\gamma_k, o_{l,k})} \right|$ and $\sum_{z^l_{(\gamma_k, o_{l,k})}, l=1,\ldots,N} \left| z^i_{(\gamma_k, o_k)*} - z^l_{(\gamma_k, o_k)*} \right|$, where $\left[z^i_{(\gamma_k, o_w)*}, z^{i\;*}_{(\gamma_k, o_w)} \right]$ is the consensus of analysed set.

We chose this algorithm because of its the lowest complexity which is linear.

4. METHOD FOR COPING WITH INCONSITENCY

There are a few different methods for coping with inconsistency. In our solution main agent, approaching inconsistency, proposes several possible results. She obtains these results through deriving separate, more consistent clusters. She examines the results and selects the most appropriate one.

4.1. DATA CLUSTERING

Let us define K as a set of clusters $(K_1, K_2, ..., K_R)$, where $R = N$ is number of observations. Suppose that $\overline{v_i} \in K_i$ for $i = 1, 2, \ldots, N$ (each vector in its own cluster). Agglomerative algorithm is defined as follows: Let be the S_u u-th step. While the number of clusters $R - u > 0$ union the closest clusters $K_j, K_p, j \neq p$, that satisfies

$$D(K) = \min_{m,n} G(K_m, K_n), m, n \in 1, 2, ..., R; m \neq n \tag{4}$$

where

$$G(K_m, K_n) = d(C^j, C^p) \tag{5}$$

is the distance defined in section 3.2., C^j, C^p are consensus vectors (defined like C in section 3.1) accordingly for K_j, K_p. The output of each step has a form $(K_1, K_2, ..., K_{R-u})$. We chose hierarchical agglomerative algorithm because it suits our consistency measuring function. Reference to wider description of that algorithm can be found in [2].

4.2. FINDING THE RIGHT AMOUNT OF CLUSTERS

For the task of fuzzy structures integration we approached clustering and measuring consistency in two differently-based ways. The first approach refers to Gap Statistic which is statistically based algorithm. Wide description of its functioning is proposed in [3]. We modified it to match our data structure. The changes regard the null reference distribution generating [1]. Since our data has a structure of bracket, we generate two null reference distribution matrixes for each $z^i_{(\gamma_k, o_{i,k})*}$ in $[z^i_{(\gamma_k, o_{i,k})*}, z^{i\,*}_{(\gamma_k, o_{i,k})}]$ for every $i = 1, 2, \ldots, N$, $k = 1, 2, \ldots, P$ and Euclidean length of

$\left|z^i_{(\gamma_k, o_{i,k})}\right|$. Afterwards we combine matching values back to form of brackets $[z^i_{(\gamma_k, o_{i,k})*}{}', z^i_{(\gamma_k, o_{i,k})*}{}' + \left|z^i_{(\gamma_k, o_{i,k})}{}'\right|]$. The rest of procedure is as in the references paper. The second method refers to Silhouette algorithm which is data structure-based, described widely in [6]. In general, silhouette algorithm assesses clustering of every data element in given set.

5. FRAMEWORK PROTOTYPE

To research further on fuzzy data integration defined in section 3 and 4, we created prototype of multi-agent framework, in which input data was generated in two different ways. The reason under such approach is to test two cases. The first case supposes that there is just one group of opinions and the inconsistency is low. The second one consists of two separate groups with low inconsistency in each of them but high inconsistency overall.

5.1. AGENTS WITHIN FRAMEWORK

In created platform agents belong to two different groups. There is one main agent and few observer agents. The task of observer agent is to observe a selected part of world and gather information about particular features of its objects. Observer agents are independent in their functioning and have no influence on each other i.e., they are cognitive. Main agent task is to question observer agents and perform the task of integrating data on gathered knowledge.

5.2. GENERATED INPUT DATA

Suppose that we observe one object on its two features. Let there be *N* observing agents and one main agent who desires to know the state of observing object. She thus asks observer agents about two defined features of the object. After the gathering, main agent conducts the integration process.

In our framework we supposed two different cases. The first case data generating is based on normal distribution with the same parameters for every of *N* observer agents. It means that agents will generate their observations within one group of low inconsistency value. The second case covers the situation of two separate groups. Suppose that within the group of fifty agents there are two different, separate groups of opinions. We generate opinions of those two different groups using normal distribution with parameters that ensure distinguishing the two groups.

5.3. DATA INTEGRATION WORKFLOW

Main agent gathers knowledge $\{\overline{O_1}, \overline{O_2}, ..., \overline{O_N}\}$ about some objects of the observed world. Gathered observations are in a form of vector containing linguistic values. She now performs the task of data integration. Before proceeding to the integration process, she is obligated to translate observations to vectors of brackets $\{\overline{v_1}, \overline{v_2}, ..., \overline{v_N}\}$. That part of the integration process requires her view on that particular part of the world implemented basing on expert's opinions. She proceeds to the actual integration process. Using Gap Statistics (described in 4.2) based on agglomerative algorithm (see Section 4.1), she determines optimized number of clusters that reflects gathered data. For comparison, she conducts the same process using Silhouette (see 4.2) instead of Gap Statistics. If the results match, main agent is assured that the result she received is the proper one. Otherwise she chooses Gap Statistics result as it is more accurate. According to the number of clusters she determined, main agent proposes a consensus $(C_1, C_2, ..., C_P)$, translated back to linguistic form, for each of the clusters that were obtained.

1.6. EXEMPLARY RESULTS

Suppose that both of two mentioned features of one object has ten possible linguistic values. We generated data sets in two different scenarios:

(a) *single-cluster data in two dimensions* – 100 observations centered around (o_1, o_2) where o_1 and o_2 are fixed linguistic values.

(b) *two clusters in two dimensions* – the clusters are (50, 50) observations centered around (o_1, o_2), (o_3, o_4), where o_1, o_2, o_3 and o_4 are fixed linguistic values, selected centres are different and distant so that there are two noticeable clusters.

We present example of realization for each of scenarios giving the gap chart.

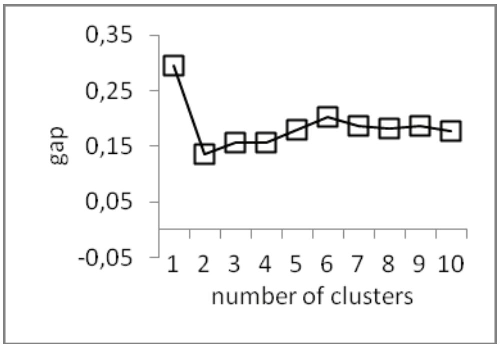


Fig. 1. Results for the first scenario generation data simulated with Gap Statistic

Scenario (a) covers the situation of uniformly distributed observation around selected centre. Figure 1 presents the results of clustering using Gap Statistic algorithm. The Gap curve has a clear maximum at 1. Result for the Silhouette are not presented because Silhouette works for at least two clusters. Obtained result matches our expectations. Figure shows gaps for up to 10 clusters because other results do not satisfy our assumption which is to minimize the number of clusters

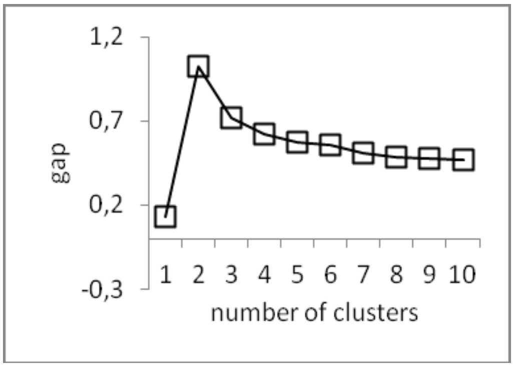


Fig. 2. Results for the second scenario generation data simulated with Gap Statistic

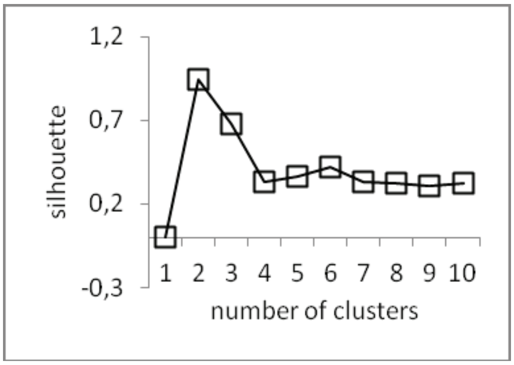


Fig. 3. Result for the second scenario generation data simulated with Silhouette

Figures 2 and 3 examine the two clusters scenario. Analyses of Fig. 2 and Fig. 3 estimate the number of clusters which is 2. The behavior of both algorithms are correct and they work according to our definitions. Figures present the results for shortened number of clusters but there is a possibility that higher number of clusters is a good solution too because there may be more less separated clusters within two well-separated.

Table 1. Results of the simulation of 20 independent realizations

Method	Estimates of the following numbers of clusters									
	1	2	3	4	5	6	7	8	9	10
1 cluster in 3 dimensions										
Gap	20	0	0	0	0	0	0	0	0	0
2 clusters in 3 dimensions										
Silhouette	–	18	2	0	0	0	0	0	0	0
Gap	0	20	0	0	0	0	0	0	0	0
4 clusters in 5 dimensions										
Silhouette	–	0	2	16	2	0	0	0	0	0
Gap	0	0	0	20	0	0	0	0	0	0

Table 1 presents results for different data cases of 20 realizations. It proves that the Gap statistic suits our data structure while Silhouette has few problems overall. Silhouette does not work for single-clustered data so it is not presented in the table.

7. CONCLUSIONS

In this paper, we have described an approach to integrate data with fuzzy structure. We successfully conducted the process of knowledge integration providing consensus for the result. Poor results in non-clustering approach were solved by using algorithms of measuring inconsistency and data clustering. Two different approaches to that problem were presented and compared. Gap Statistic proved to be better than Silhouette and able to solve more difficult problems along with single-clustering scenario. Although we experiment only with few different cases the solution can be easily applied to wider problems and different clustering.

We plan to approach static compatibility functions with similar methods to those described in this paper. Framework based on multi-agent system can be approach in many different ways. We described only one view with main agent in the centre and observer agents gathering data. The scenario of agents communicating with each other is valuable and worth describing.

ACKNOWLEDGEMENTS

This paper was supported by Grant no. N N519 444939 funded by Polish Ministry of Science and Higher Education (2010–2013).

REFERENCES

[1] GORDON A., *Null models in cluster validation*, [in:] W. Gaul, D. Pfeifer (eds.), *From Data From Knowledge*, Springer, New York, 1996, 32–44.

[2] HAND D., MANNILA H., SMYTH., *Grupowanie Hierarchiczne*, [in:] *Eksploracja Danych*, WNT, Warszawa, 356–362.
[3] KULKARNI A.K., PEDERSEN T., *Unsupervised Context Discrimination and Automatic Cluster Stopping*, Minnesota, 2006, 28–32.
[4] NGUYEN N.T., *Advanced Methods for Inconsistent Knowledge Management*, Springer-Verlag, London, 2008, 47–100.
[5] NGUYEN N.T., *Consensus Methods and Their Applications to Solving Conflicts in Distributed Systems*, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, 24, 2002, 112–129.
[6] ROUSSEEUW P.J., *Silhouettes: A graphical aid to the interpretation and validation of cluster analysis*, J. Comput. Appl. Math., Vol. 20, No. 1, 1987, 53–65.
[7] TIBSHIRANI R., WALTHER G., HASTIE T., *Estimating the number of clusters in data set via the gap statistic*, Stanford University, USA, 2000, 411–423.
[8] ZADEH L.A., *The concept of Linguistic Variable and its Application to Approximate Reasoning-I*, Information Sciences, Vol. 8, 1975, 199–249.

consensus, knowledge integration,
multi agent systems

Grzegorz SKORUPA, Jan JAKUBIK,
Bartosz FIDRYSIAK*

A CONSENSUS METHOD FOR INTEGRATION OF KNOWLEDGE EXPRESSED IN FORM OF MODAL FORMULAS

In this paper, we propose an approach to the problem of integration of knowledge expressed in form of modal formulas with operators for possibility, belief and certainty. We assume that an agent is collecting messages describing properties of an external object from a group of other agents in order to create his own prepositional attitude. The language of communication allows sentences in form OP(p), where OP is a modal operator of possibility, belief or certainty, and p is logical conjunction, alternative or exclusive alternative of two prepositional variables. The receiving agent collects the knowledge using a form of internal representation. A simple consensus method is applied in order to integrate the gathered knowledge, and the resulting consensus is subsequently used to create a set of sentences that can be communicated by the agent, according to the rules described in works concerning the problem of grounding modalities [1]. We developed a simple framework for testing this algorithm and present sample results of integration.

1. INTRODUCTION

The need for integration of agents' prepositional attitudes is directly connected with incompleteness and inconsistency of knowledge in multi agent systems. The incompleteness results from limited agents' cognitive abilities [2] that allow them to gather only partial knowledge about their external world [5]. The ability of gathering valid information describing the world by each agent depends on individual degree of precision and influence of environment. It often leads to large variety in knowledge of agents. Integration of knowledge requires all agents to cooperate with each other in order to find a reasonable consensus.

* Institute of Informatics, Wroclaw University of Technology, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław.

In this paper we present a prototype of multi agent platform that simulates generating sentences and finding a consensus for a communication language [6] based on modalities such as „it possible that” and „I believe that”. The structure of knowledge and the way of generating sentences is based on other works concerning the problem of language grounding [4].

Our approach to the integration of modal formulas is similar to one that has been already proposed in other works [3]. However, in the cited work, a consensus method was used to find a single modal sentence that is the best approximation of collected inconsistent knowledge. This was done by translating the sentences to an internal representation of knowledge, finding consensus for this internal representation and then choosing appropriate sentence. In this paper, we aim to choose a set of sentences that can be communicated by the agent performing the integration task, instead of a single sentence. The reasoning behind that approach is that, given the limits of the communication language, single sentence is only a part of description of the world, as opposed to a set of sentences. We assume that the consensus found using the internal representation approximates the state of the world observed by agents, and we derive a set of messages from the consensus, following rules for grounding of modalities [4].

In section 2 we describe knowledge integration task. Section 3 holds definition of representation of knowledge and algorithm of finding a consensus. In turn, in further sections there is presented a multi agent platform that generates a sample data and finds a consensus.

2. KNOWLEDGE INTEGRATION TASK

Knowledge integration is a common problem in multi agent systems in which knowledge inconsistency needs to be resolved. While single agent’s prepositional attitude can be based on his own observations and therefore should be consistent, these observations may be affected by conditions that vary for different agents, such as position of observer, time of performing observation and interferences of environment. This means that in most cases, knowledge gathered from a group of agents will not be consistent and should be integrated.

The basic assumption in our work is that an agent is collecting data from external sources in order to create her prepositional attitude. However, data sent to that agent is not a full description of other agents’ states of knowledge. A modal language of communication is used, and the knowledge is gathered in form of sentences that express communicating agents’ beliefs concerning the observed object.

In order to resolve inconsistency, we create an internal representation of knowledge, which approximates the underlying meaning of modal formulas used in communication. On level of internal representation, a consensus is easily found and subsequently can be translated to the language of communication. The reasoning behind this kind of approach has been explained in [3].

2.1. DEFINITION OF CONSENSUS

Consensus methods are used in order to resolve knowledge conflicts and analyze inconsistent data [7]. The analyzed dataset for a consensus method represents a conflict in which different participants represent different opinions about the same subject. Consensus methods are employed in order to find a data record that is the best representation of an inconsistent set of records, which represent the conflict. In typical consensus-based approach, a profile composed of elements of the same type is given. The consensus algorithm requires definition of a distance function for the particular type of data and aims to find an element that minimizes the sum of distances or squared distances to the elements in the profile. Multiple postulates a proper consensus should satisfy have been defined in [7].

3. FINDING CONSENSUS

3.1. KNOWLEDGE REPRESENTATION

Let us define a representation of knowledge. The representation of knowledge was previously used in other algorithms [3]. Suppose, that each agent observing the world, asked for a considered object, generates a set of complex sentences. Every complex sentence has the following form: $OP(\varphi)$ where $OP \in \{Pos, Bel, Know\}$ and $\varphi \in \{p \wedge q, p \wedge \neg q, \neg p \wedge q, \neg p \wedge \neg q, p \vee q, p \underline{\vee} q\}$. Particular operators ($OP$) mean respectively possibility, belief and knowledge that sentence is true according to an agent. $\varphi = p \wedge q$ means that both p and q properly describe the object in opinion of an agent (other formulas are explained by analogy) where p and q are some properties of a fixed object. $Bel(p \wedge q)$ has semantic meanings "I believe that object has property p and q". Formal syntax and semantics were defined in [4]. Semantic definitions are consistent with Hintikka's works [1]. Suppose that there exists a central agent who gathers sentences from all agents. The gathered sentences create together a M-element set K. Each i-th sentence has it's internal representation as a tuple (a_i, b_i, c_i, d_i) where particular elements mean respectively in what degree formulas $p \wedge q$, $p \wedge \neg q$, $\neg p \wedge q$, $\neg p \wedge \neg q$ describe the object according to an agent. The internal representation of knowledge needs to satisfy the following conditions:

- $$\forall i \in [1, M] \;\; a_i + b_i + c_i + d_i \le 1 \tag{1}$$

- $$\forall i \in [1, M],\ 0 \le a_i \le 1,\ 0 \le b_i \le 1, 0 \le c_i \le 1, 0 \le d_i \le 1 \tag{2}$$

The function $rep(OP(\varphi)) = (a_i, b_i, c_i, d_i)$ which assigns an internal representation to a sentence is defined as:

$$\begin{gathered}
rep(OP(p \wedge q)) = (val(OP),0,0,0) \\
rep(OP(p \wedge q)) = (val(OP),0,0,0) \\
rep(OP(p \wedge \neg q)) = (0, val(OP),0,0) \\
rep(OP(\neg p \wedge \neg q)) = (0,0,0, val(OP)) \\
rep(OP(p \vee q)) = \left(\frac{val(OP)}{3}, \frac{val(OP)}{3}, \frac{val(OP)}{3}, 0 \right) \\
rep(OP(p \underline{\vee} q)) = \left(0, \frac{val(OP)}{2}, \frac{val(OP)}{2}, 0 \right)
\end{gathered} \tag{3}$$

Function *val* sets conviction with which an agent speaks a sentence . The signature of the function looks as follows $val : OP \rightarrow [0, 1]$. Values of particular operators are arbitrarily set.

3.2. METHOD FOR DATA INTEGRATION

The central agent gathers data from all agents. It receives all elements of a set *K*. Then, it uses a representation function to translate each sentence $OP(\varphi)$ to a tuple (a_i, b_i, c_i, d_i). All translated sentences are stored in a set *SK* which is also a *M*-element set. Having set *SK*, the central agent can start computing a consensus.

In order to find a consensus on the semantic level, we need to find a consensus of profile of tuples. For a distance measure, we use four-dimensional Euclidean distance:

$$\begin{gathered}
d((a_i, b_i, c_i, d_i), (a_j, b_j, c_j, d_j)) \\
= \sqrt{(a_i - a_j)^2 + (b_i - b_j)^2 + (c_i - c_j)^2 + (d_i - d_j)^2}
\end{gathered} \tag{4}$$

In order to find a consensus of the profile *P* we use the following formula:

$$C(P) = (a,b,c,d) = \left(\frac{1}{M} \sum_{i=1}^{M} a_i, \frac{1}{M} \sum_{i=1}^{M} b_i, \frac{1}{M} \sum_{i=1}^{M} c_i, \frac{1}{M} \sum_{i=1}^{M} d_i \right) \tag{5}$$

The result minimizes sum of squared distances, which can be simply proven by showing that:

$$\sum_{i=1}^{M} d(C(P),(a_i,b_i,c_i,d_i))^2$$

$$=\sum_{i=1}^{M} \sqrt{\left(a-a_j\right)^2+\left(b-b_j\right)^2+\left(c-c_j\right)^2+\left(d-d_j\right)^2}^{\,2} \tag{6}$$

$$=\sum_{i=1}^{M}(a-a_i)^2+\sum_{i=1}^{M}(b-b_i)^2+\sum_{i=1}^{M}(c-c_i)^2+\sum_{i=1}^{M}(d-d_i)^2.$$

It has been shown that arithmetic mean minimizes the sum of squared distances and can be used as consensus for profile of numbers [7]. Therefore the presented formula (5) minimizes each of the four addends in the sum (6).

3.3. CREATING LINGUISTIC REPRESENTATION OF CONSENSUS

In order to create a full linguistic representation, we need a four-element tuple that satisfies the following conditions:

- $$a+b+c+d=1 \tag{7}$$
- $$0\le a\le 1,\ 0\le b\le 1,\ 0\le c\le 1,\ 0\le d\le 1 \tag{8}$$

Note that the condition (8) should be already met by consensus C because the same condition was assumed for the internal representation of a single sentence (2). In order to derive a set of sentences that can be communicated by the receiving agent, we create a new tuple:

$$(a',b',c',d')=\left(\frac{a}{a+b+c+d},\frac{b}{a+b+c+d},\frac{c}{a+b+c+d},\frac{d}{a+b+c+d}\right) \tag{9}$$

The result meets the criterion (7) and therefore can be translated to a set of sentences according to the rules of grounding modal conjunctions and alternatives [4]. Using rules given from [4] we created an algorithm generating a set of sentences based on arbitrary set values for $\min_{\text{pos}}$, $\min_{\text{bel}}$, $\min_{\text{bel}}$, $\max_{\text{bel}}$, $\min_{\text{know}}$, and e

Algorithm 1

$\min_{\text{pos}} = 0{,}1$	– threshold of minimum value of operator *Pos*
$\max_{\text{pos}} = 0{,}6$	– threshold of maximum value of operator *Pos*
$\min_{\text{bel}} = 0{,}6$	– threshold of minimum value of operator *Bel*
$\max_{\text{bel}} = 1$	– threshold of maximum value of operator *Bel*
$\min_{\text{know}} = 1$	– threshold of minimum value of operator *Know*
$e = 0{,}1$	– a value from partition (0, 1]

Values of parameters can be changed. Values given above were used in framework described in paragraph 4.

1. Create an empty result set.
2. If $a' \geq \min_{\text{pos}}$, add $OP(p \wedge q)$ to the result, where *OP* = *Pos* if $a' \in [\min_{\text{pos}}, \max_{\text{pos}})$, *OP* = *Bel* if $a' \in [\min_{\text{bel}}, \max_{\text{bel}})$ and *OP* = *Know* if $a' \geq \min_{\text{know}}$.
3. If $b' \geq \min_{\text{pos}}$, add $OP(p \wedge \neg q)$ to the result, where *OP* = *Pos* if $b' \in [\min_{\text{pos}}, \max_{\text{pos}})$, *OP* = *Bel* if $b' \in [\min_{\text{bel}}, \max_{\text{bel}})$ and *OP* = *Know* if $b' \geq \min_{\text{know}}$.
4. If $c' \geq \min_{\text{pos}}$, add $OP(\neg p \wedge q)$ to the result, where *OP* = *Pos* if $c' \in [\min_{\text{pos}}, \max_{\text{pos}})$, *OP* = *Bel* if $c' \in [\min_{\text{bel}}, \max_{\text{bel}})$ and *OP* = *Know* if $c' \geq \min_{\text{know}}$.
5. If $d' \geq min_{\text{pos}}$, add $OP(\neg p \wedge \neg q)$ to the result, where *OP* = *Pos* if $d' \in [\min_{\text{pos}}, \max_{\text{pos}})$, *OP* = *Bel* if $d' \in [\min_{\text{bel}}, \max_{\text{bel}})$ and *OP* = *Know* if $d' \geq \min_{\text{know}}$.
6. If $Pos(p \wedge q), Pos(p \wedge \neg q)$ and $Pos(\neg p \wedge q)$ are already in the result and $(|a' - b'| < e) \wedge (|a' - c'| < e) \wedge (|b' - c'| < e)$, add $OP(p \vee q)$ to the result, where *OP*

= *Pos* if $a' + b' + c' \in [\min_{\text{pos}}, \max_{\text{pos}})$, $OP = Bel$ if $a' + b' + c' \in [\min_{\text{bel}}, \max_{\text{bel}})$ and $OP = Know$ if $a' + b' + c' > \min_{\text{know}}$.

7. If $Pos(p \wedge \neg q)$ and $Pos(\neg p \wedge q)$ are already in the result, $OP(p \vee q)$ was not added to the result for any *OP* and $|b' - c'| < e$, add $OP(p \veebar q)$ to the result, where $OP = Pos$ if $b' + c' \in [\min_{\text{pos}}, \max_{\text{pos}})$, $OP = Bel$ if $b' + c' \in [\min_{\text{bel}}, \max_{\text{bel}})$ and $OP = Know$ if $b' + c' > \min_{\text{know}}$.

In [4] it has been shown that by setting the values of $\min_{\text{pos}}$, $\max_{\text{pos}}$, $\min_{\text{bel}}$, $\max_{\text{bel}}$, $\min_{\text{know}}$, and e we can limit the number of possible message sets, i.e. if $\min_{\text{bel}} > 0{,}5$ then it is impossible to express two contrary beliefs (because no tuple that meets criteria (7) and (8) can be translated to a set that contains two or more sentences with *Bel* operator).

Algorithm 2
Now, let us present the algorithm of integration:
1. Gather information of the chosen object and put it to set *K*.
2. Translate each sentence from *K* separately to the internal representation of knowledge using the function of representation (3) and put result to set *SK*.
3. Compute the consensus of set *SK* in form of an internal representation according to the formula (5).
4. Translate consensus to external representation in form of a set of sentences using algorithm 1.

4. FRAMEWORK PROTOTYPE

We created a multi-agent program realizing the task of integration described in paragraph 3. The data was generated randomly and then it was respectively hummed. In further paragraphs, we present a role of agents and a method of generating a data.

4.1. AGENTS WITHIN FRAMEWORK

The task of each agent is to observe the concrete object that is a part of the external world and make a personal attitude to her. Agents are independent and do not communicate with each other. Every single agent generates a set of sentences that describe the object best according to her. The structure of generated sentences is precisely described in paragraph 3.1. The central agent gathers sentences from all agents and translates them to the internal representation. Then, it is computing the consensus using the formula (5).

4.2. GENERATED INPUT DATA

Before description of generating sentences, the real state of considered object needs to be defined. The real state of object is a tuple $(\alpha, \beta, \chi, \delta)$ where particular elements mean respectively in what degree formulas $p \wedge q$, $p \wedge \neg q$,$\neg p \wedge q$, $\neg p \wedge \neg q$ describe the considered object. Generating a set of sentences, by each agent, process in 2 phases:

- creating a tuple by humming the tuple of the real state of object.
- translating the obtained tuple to external representation. It is described by algorithm 1.

The process of humming relies on random changing the particular elements of a tuple so that each element of the tuple would not be changed by the value greater that earlier specified hum. What is more, after humming, conditions (7) and (8) cannot be shaken.

4.3. DATA INTEGRATION WORKFLOW

Below we present a data integration workflow:

1. Generating hummed tuples by each agent according to the process of humming described in paragraph 4.2.
2. Translating every hummed tuple to it’s external representation using algorithm 1 and put the result to set *K*.
3. Computing a consensus according to algorithm 2.

In algorithm 2 we use following values for internal representation of knowledge (3):

$$val(Pos) = 0{,}3\,;\ val(Bel) = 0{,}7\,; val(Know) = 1 \qquad (10)$$

5. EXEMPLARY RESULTS

Let us consider an object which is observed by 120 agents. Suppose that $hum = 0{,}2$. The agents generate sentences by humming a tuple (0,2; 0,7; 0,1; 0). There are presented exemplary hummed tuples, sentences said by particular agents and computed consensus below.

Agent 1

Hummed tuple: (0,2595; 0,6275; 0,0897; 0,0233)

Sentences: $\{ Pos(p \wedge q), Pos(p \wedge \neg q) \}$

Agent 2

Hummed tuple: (0,1022; 0,7320; 0,1658; 0)

Sentences: $\{ Pos(p \wedge q), Bel(p \wedge \neg q), Pos(\neg p \wedge q) \}$

Consensus: $\{ Pos(p \wedge q), Bel(p \wedge \neg q), Pos(\neg p \wedge q) \}$

All generated tuples satisfy the required conditions mentioned in earlier paragraphs. The obtained sentences are translated properly. To make sure the received consensus is logical, we should compare it to a set of sentences created by translating tuple (0,2. 0,7; 0,1; 0) to an external representation according to algorithm 1. The result of the translation looks as follows: $\{ Pos(p \wedge q)$, $Bel(p \wedge \neg q)$, $Pos(\neg p \wedge q) \}$. As we can see, the translation of the tuple is the same as the received consensus what provides us the consensus is consistent with assumed real values.

6. CONCLUSIONS

We presented an algorithm for integration of inconsistent knowledge expressed in language of modal formulas. The algorithm is based on an existing consensus method that employs the idea of language grounding. Sentences received in messages are translated to the internal representation of knowledge, and a consensus is found on the semantic level. As opposed to the existing method, in which a single sentence that could be considered the best representation of the sentences profile was chosen, the consensus calculated on semantic level is here used to create new mental model of the world that can be used by the receiving agent to communicate new messages, derived from it using the concept of language grounding. Reasoning behind this approach is that single sentence is only a partial description of the agent's prepositional attitude.

In future works this idea can be further developed. An agent could attempt to recreate the state of the world for every other agent it is communicating with. Another approach to the integration of modal formulas can be based on integration of formula sets received from different agents.

ACKNOWLEDGEMENTS

This paper was supported by Grant no. N N519 444939 funded by Polish Ministry of Science and Higher Education (2010–2013).

REFERENCES

[1] HINTIKKA J., *Knowledge and belief. In: An introduction to the logic of the two notions*, Cornell University Press, 1962.
[2] HUHNS N., SIGNH M., *Cognitive Agents*, IEEE Internet Computing, 2(6), 1998, 87–89.
[3] KATARZYNIAK R., PIECZYŃSKA-KUCHTIAK A., *An approach to resolving semantic inconsistency of multiple prepositional attitudes*, Journal of Intelligent & Fuzzy Systems, Vol. 17, No. 3, 2006, 231–238.
[4] KATARZYNIAK R., *Grounding of modal communication languages in artificial agents*, EXIT Warsaw, Poland, 2005.
[5] KATARZYNIAK R., NGUYEN N.T., *Reconciling inconsistent profiles of agents' knowledge states in distributed multi-agent systems using consensus methods*, System Science 26(4), 2000, 93–119.
[6] KATARZYNIAK R., *The Language Grounding Problem and its Relation to the Internal Structure of Cognitive Agents*, Journal of Universal Computer Science **11**(2), 2005, 357–374.
[7] NGUYEN N.T., *Advanced Methods for Inconsistent Knowledge Management*, Springer-Verlag, London, 2008.

consensus, clustering,
integration, relational, sensor

Dominik WIĘCEK, Aleksander ŚWIĄTKOWSKI,
Łukasz LEŚNICZEK*

MULTI-AGENT FRAMEWORK FOR INCONSISTENT SENSOR DATA INTEGRATION

Paper presents a framework for a system collecting information from different sources (physical sensors). Data gathered this way is inconsistent and incomplete. We are integrating data into a consistent set of easily accessible and manageable records. The main process is based on consensus algorithm which converts data using summation minimization. Principal agent not only integrates data, but checks whether the level of consistency in the data set is acceptable or not and may propose alternative solutions with higher level of consistency. To achieve this task agent uses clustering methods. We implemented a prototype platform performing described integration and present achieved results.

1. INTRODUCTION

Let us suppose that we are working with data gathered from a number of sources, regarding the same subject. If the method of data gathering of the sources differs, it leads to the problem of the sources contradicting each other. The origin of inconsistencies in such models may be the sources' autonomy, data corruption etc. There emerges a need to determine which parts of the data retrieved is truly authentic, and which is false and should be rejected. Data integration task is a process of extracting a *consensus* out of such a set of data, that is otherwise contradictory [1].

Data integration issue has already been explored. Methods based on consensus algorithm are commonly used to solve integration task [2]. A *consensus* is defined as an output set of data that is consistent and includes one complete description of every object that was described in a several ways in the input data set.

* Institute of Informatics, Wrocław University of Technology, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław.

In this paper, we will focus on the relational model, so the integration will be performed on a set of tuples. Each tuple represents features of real world objects. The goal of the integration task [5] is to integrate several, often contradictory, tuples into one which can be objectively assumed to be the most probable. The result of this process is a set of tuples which should consistently describe current state of the world and its objects.

We present a prototype multi-agent framework implementing the integration task. The framework consists of several modules. The first one is the data generator, which creates the required tuples. The second is the module that gives a set of tuples, then checks their consistency rate and may proposes alternative sets created with clustering algorithm. The third module constructs a consensus for each of the groups, which is a single tuple with each of its attributes stimulated on the ground of the tuples in the group. The input data is generated such that it's initially roughly grouped. The results show that our algorithm can successfully identify the clusters and extract a consensus for each of them. We recreate the problem by simulating a physical herd of cows, and sensors gathering their data. Each cow is being observed by at least a handful of sensors, every one having a chance of being flawed and observing the cow's features wrong. This way, we receive data that is corrupted in a way that allows for the interesting results.

In our solution we not only integrate data, but additionally we assess level of consistency in the result set. Furthermore if level of consistency is to low we will propose alternative solutions with higher consistency rate. We achieved that by using a clustering algorithm which helps to group data into more consistent sets. The solution can be used to come up with possible alternate situations in any environment where data is gathered in a way similar to the described above.

In the second paragraph described in details is the knowledge integration task, including the definition of consensus. How to determine if the data is really inconsistent. In the third paragraph, we show how to find a consensus in various situations. The fourth paragraph deals with the methods of coping with inconsistent knowledge. Data clustering algorithms are presented. The fifth paragraph contains a description of our framework and how it works. The last two paragraphs contain results and conclusions of our study.

2. KNOWLEDGE INTEGRATION TASK

Wherever there is a system that depends on data retrieved from a number of physical sources, which can malfunction, there is a need for data integration. According to Reimer [10] knowledge integration can be described as eliminating inconsistency appearing in a data set. In systems which model existing objects, possessing numerous features, it is of utmost importance that the data is put together in a way allowing the greatest possible reliability.

The goal of data integration is to take a multiset containing inconsistent data and come up with a subset containing clearly defined beforehand, consistent data. Usually, the abovementioned multiset would contain several different representations of a single object, and we would want to fuse them together to end up with a single one. So the integration task is, in layman's terms, like negotiating a common point of view among a group of people.

In this paper, we are interested in integrating tuples with a fixed number of attributes. Every object will be initially described by some number of tuples, which may or may not be identical to one another. Each tuple represents one sensor's observations of an object. The data may be incomplete or corrupted in some aspects. Our goal is (for now) to end up with a single tuple for every object, which is a result of integration of the data received from sensors. Later on, we'll be able to group the tuples into clusters, and then integrate, coming up with several alternative consensuses.

2.1. FINDING CONSENSUS

According to Nguyen [7] the subject of a consensus method is a set containing different versions of a knowledge about real word objects. The main goal of consensus function is to determine which version of knowledge describes real world most precisely. There are a lot of quite different ways to achieve this task. There are algorithms based on summation minimization or algorithms which minimize the sum of squared distances between information about objects. These are used in different cases which depend on the type of data structure integrated and distance function chosen.

2.2. HOW HUMAN COPES WITH INCONSITENCY OF KNOWLEDGE

When we encounter inconsistent data in real life, we have a lot of different ways to deal with it. However, human actions are unpredictable and tough to simulate by computer programs. A common and the simplest algorithm that tries to give results similar to human propositions is arithmetic mean. It should work in most cases where we deal with real numbers or integers. Where there is an attribute with a limited number of values, then probably the most common value should be considered the consensus. When dealing with a more complicated case, humans have developed multiple cognitive strategies to solve the problem. For example they could reject some data, probably that which reduces the consistency level, qualifying it as corrupted or erroneous and thus useless. Or when the data seems to be roughly grouped, then the most natural solution would be to separate it, and treat independently, to receive alternative solutions. Particular items of the set can also be evaluated, based on the knowledge source reliability. Our framework tries to simulate some of the mentioned human behaviours to solve the inconsistent data problem.

Agents [8], [9] can be used to simulate human behaviour. In multi-agent frameworks, agents can be used to effectively simulate the so-called *collective consciousness* – which has been proven to successfully take on tasks more difficult than any of the members of the collective ever would. Programming an agent to behave as such as it had some kind of intuition, typical for human beings, is certainly not trivial. It seems that a number of agents, each with its own, individual "beliefs", will realistically model some of the human decisions taken in such situations [3].

3. FINDING CONSENSUS

Our knowledge structure is described by the following set (***A***, ***V***, ***O***, ***S***, ***R***), where:

$A = \{A_1, A_2, ..., A_M\}$ – set of attributes,

$V = \{V_1, V_2, ..., V_M\}$ – set of sets of attributes values,

$O = \{o_1, o_2, ..., o_N\}$ – set of objects,

$S = \{s_1, s_2, ..., s_U\}$ – set of sensors,

$R = \{r_1, r_2, ..., r_Z\}$, where every r_z is a a single observation in the form of

$r_z = \{s_u, o_n, v_{1,z}, v_{2,z}, ..., v_{M,z}\}$, $v_{i,z} \in V_i \cup \{null\}$; $i = 1, 2, ..., M$; $u = 1, 2, ..., U$; $n = 1, 2, ..., N$.

A holds various attributes of the observed objects. V is a set of sets of attributes values, which means that the attribute A_m is limited to taking values from the set V_m. O is a set of observed objects. S is a set of sensors observing objects. R represents all the data that has been gathered and will be later integrated. Every r_z is an observation regarding a single object made by a single sensor. The observations include objects' features as perceived by sensors. Each of the $v_{z,m}$ in every r_z is the value of the m-th attribute in the z-th observation.

Additionally we define helper selection functions for observation r_z:

a) $H_A : R \times A \to V_1 \cup V_2 \cup ... \cup V_M \cup \{null\}$, such that $H_A(r_z, A_m) = v_{m,z}$

b) $H_O : R \to O$, such that $H_O(r_z) = o_n$

Function H_A returns value of attribute A_m in observation r_z. Function H_O returns observed object for observation r_z.

Additionally, the *null* value was introduced to describe the situation when we don't have any information about particular feature. Appearance of this symbol indicates that one of the agents couldn't gather some data about the object.

The principal agent takes care of the integration. It is assumed that the objects are recognizable by a unique key o_n. The integration process is carried out separately for every object. The resulting set takes the same structure as the input data *R,* except that it is consistent.

3.1. METHOD FOR DATA INTEGRATION

Every attribute has its own distance function. It is advised that each of the distance function is normalised. Additionally, if in one of the tuples attribute has value *null* and in the other the value is known, then the distance between this attributes should be equal to one. If two the same attributes are *null* the distance between them should be zero. Let d be the distance function for the whole observation, and let d_m be the distance function for attribute m. Then:

$$d(r_i, r_j) = d_1(v_{1,i}, v_{1,j}) + d_2(v_{2,i}, v_{2,j}) + \dots + d_M(v_{M,i}, v_{M,j}) \tag{1}$$

where we assume that for $m = 1, 2, \dots, M$:

a) $d_m : V_m \times V_m \to [0;1]$

b) $d_m(null, x) = 1$, where $x \in V_m$ and $x \neq null$

c) $d_m(null, null) = 0$

Using chosen distance function we define consensus function. Let V_n^m be the set of values of attribute A_m collected from all sensors, that observed object n:

$$V_n^m = \{v_{m,z} : \exists r_z \in R : H_A(r_z, A_m) = (v_{m,z}) \wedge H_O(r_z) = o_n\}$$

Then consensus function will be:

$$c(V_n^m) = \underset{v^* \in V_n^m}{\arg\min} \sum_{v \in V_n^m} d(v^*, v) \tag{2}$$

The definition means that for each set of values set of some attribute function c will choose set of values that minimize the sum of distances to all observed values. We adapted the algorithm to our knowledge structure and added a module responsible for checking the level of consistency in result set and for suggesting alternative consensuses by using clustering algorithm. Computational complexity of our algorithm is linear and depends on number of objects, sensors and attributes. The result is a set C^* of L alternative consensuses, each containing N observations, such that:

$$C^* = \{C_1, C_2, ..., C_L\},$$
$$C_i = \{c_1^i, c_2^i, ..., c_N^i\}, i = 1, 2, ..., L.$$

where c_n^i, $n = 1, 2, ..., N$ is a generated observation consensus for n-th object and i-th consensus. Consensus is c_n^i of the form:

$$c_n^i = \{o_n, v_{1,i}, v_{2,i}, ..., v_{M,i}\}, \quad \text{where} \quad v_{m,i} \in V_m \cup \{null\}, m = 1, 2, ..., M$$

Each value $v_{m,i}$ is found according to formula (2) so $v_{m,i} = c(V_n^m)$.

To simplify the notation, if the algorithm identified a single cluster, then the result will simply be a set C of N observations, defined as:

$$C = \{c_1, c_2, ..., c_N\}.$$

4. METHOD FOR COPING WITH INCONSITENCY

If the input data is classified as inconsistent, the principal agent comes up with alternate solutions. The sensors are grouped by the principal agent into several clusters consisting of sensors with similar observations. The data from each cluster's sensors is integrated separately, completely rejecting the data from the other ones. The results of separate integrations are then presented as alternate solutions to the integration task. For the clustering to have a point, the data should come from sources that present radically different data extraction techniques. This way, the data will be initially roughly divided into disjoint classes. Alternate solutions for each object would make the outcome bloated and incomprehensible, by forcing the user to bear in mind different combinations of the alternatives. Instead, the algorithm presents alternative solutions for the database as whole.

4.1. DATA CLUSTERING

To effectively cluster the input data R, we decided to choose the K-means algorithm [6]. It is easy to implement, and requires employing distance functions. We use separate distance functions for each of the attributes, summing them up to produce the final distance between two given tuples. The maximum distance between two values of an attribute is 1. The maximum distance between two tuples is equal to the number of attributes. The formula for calculating the distance is the same as (1). The result of clustering is a set of clusters: $\{R_1^n, R_2^n, ..., R_K^n\}$, where n is an object index. The clusters are generated for every object. The clustering is carried out for each object separately.

4.2. MEASURING RESULT CONSISTENCY

To determine the optimal number of clusters, we implemented the cross algorithm [4]. The clusters need to group similar elements together, yet be distinct from each other. The cross algorithm that we chose divides the input set into an increasing number of clusters, and checks the division's precision. The measure of precision was the mean distance from the centers of respective clusters. The initial cluster centers are chosen at random from among every cluster's items. Then the algorithm groups the items with K-means algorithm.

The algorithm continues to divide the initial set until the difference between precision levels in subsequent steps goes below a given threshold. Let p be given threshold, f_{k-1} be mean distance from the centers of previous clusters and f_k be mean distance from the centers of actual clusters, then algorithm stops if:

$$p > \frac{f_{k-1} - f_k}{f_{k-1}}.$$

This way, we determine the optimal number of clusters that the data needs to be separated into.

5. FRAMEWORK PROTOTYPE

A program completing the integration task defined in 3. and 4. was implemented. The theme of the project was a simulated cow farm. It was assumed that there are numerous sensors scattered across the field on which the cows reside. The sensors are activated simultaneously, and observe certain features of cows in a limited range. The observations are then sent to the principle agent as the input data. Within our simulation, the data representing the actual features of the cows is stored separately, and can be later used to measure the efficiency of the algorithm. Furthermore, we assumed that some of the sensors are broken and may corrupt any or all of the values in their observations. This was implemented to make the input data set inconsistent and contradictory.

Referring to the structure described in 3.1, the relevant sets are:

$S = \{1, 2, 3, ..., S\}$, where S is the number of sensors,

$O = \{1, 2, 3, ..., N\}$, where N is the number of cows,

$A = \{$sex, localization, position, movement, action, age, temperature$\}$,

$V = \{V_{sex}, V_{lokalization}, V_{position}, V_{movement}, V_{action}, V_{age}, V_{temperature}\}$,

where:

$V_{sex} = \{ \text{f}, \text{m} \}$

$V_{localization} = X \times X$, cartesian product, describing geographic localization of each cow. It's assumed that the farm is 10000 square metres (100 m × 100 m) big.

$V_{position}$ = {n, ne, e, se, s, sw, w, nw},
$V_{movement}$ = {walks, stands, lies},
V_{action} = {eats, chews, drinks, sleeps, nothing},
V_{age} = {calf, heifer, cow, young bull, bull},
$V_{temperature}$ = {low, normal, high}.

5.1. AGENTS WITHIN FRAMEWORK

Within the framework, two types of agents have been defined. The first type is called a sensor. It comes in two versions – functional and broken. The main task which has been assigned to this agent is to observe, digitalize, and output features of all objects within its range. It is assumed that all the sensors are activated at once, scanning their part of the farm. Then they send their observations to the principle agent, described later, in the form described in 3.1. The functional sensors perceive the features just as they are, without any disturbance. The broken sensors (which don't necessarily have to be broken, they're here to simulate errors made by insufficient precision of measurements, blurry image, ambiguous input etc.) perceive the reality in a distorted way, which causes them to observe a value differing from the actual state. The sensors are widely spread across the farm, and it was made sure that at least a few sensors observe each cow.

The second type of agent is called the principal agent. Its function is to perform the data integration in the way described in 3, 4.1 and 4.2 on the data received from sensors, then output the result to a user.

5.2. GENERATED INPUT DATA

The environment in which objects (cows) exist, is a farm, a grid of 100 × 100 cells in our simulation. The sensors are spread all across the field. The actual state is generated randomly with different probability distributions for every attribute of a cow. Then, the sensors are "activated". They scan for objects within their range, and store their features in their own temporal memory. At this stage, if the sensor is broken, the data will become corrupted. Then the data from every sensor is sent to the principal agent.

5.3. DATA INTEGRATION WORKFLOW

When the principal agent receives the data, it begins integrating. First, it rates the initial set's level of consistency, and determines the number of clusters, and therefore the number of alternate solutions (described in 4). Then, for each object, the agent creates a set containing only the tuples describing that specific object, and divides it to as many parts as it was determined earlier, thus clustering data. Each cluster is treated separately from now on. The integration proceeds according to the algorithm described in 3.2. Every cluster ends up containing one tuple for each cow. The clusters are presented as the final result of integration.

6. EXEMPLARY RESULTS

We present exemplary results produced by our framework. First, our platform generated data about the entire cow herd creating structure described in 3.1. We chose and presented only a small piece of this data. Information regards cow 64. Sensors 0, 1 and 11 have a defect so their observations gave false results:

r_1 = {0, 64, f, 15, 14, n, walks, drinks, cow, normal},
r_2 = {1, 64, f, 16, 13, ne, walks, drinks, heifer, normal},
r_3 = {10, 64, m, 6, 6, s, lies, sleeps, heifer, normal},
r_4 = {11, 64, f, 15, 15, n, lies, drinks, cow, normal},
r_5 = {12, 64, m, 8, 4, s, null, sleeps, heifer, normal}.

When principal agent doesn't use the clustering algorithm, it integrates the data and returns a single consensus C = {64, *F*, 12, 10, *N*, walks, heifer, normal}.

Information that we can read from the tuple doesn't describe the cow 64 very well, because her real features are different. Additionally level of consistency in input data set is low. In this situation the principal agent may propose alternative results using *k*-means clustering. Agent decided that two clusters will be enough, so she grouped input data set into two separate clusters:

R_1^{64} = {{0, 64, f, 15, 14, n, walks, drinks, cow, normal},
{1, 64, f, 16, 13, ne, walks, drinks, heifer, normal},
{11, 64, f, 15, 15, n, lies, drinks, cow, normal}}.

R_2^{64} = {{10, 64, m, 6, 6, s, lies, sleeps, heifer, normal},
{12, 64, m, 8, 4, s, null, sleeps, heifer, normal}}.

Then for each cluster separately principal agent ran the integration algorithm. For this example two alternative consensuses look like:

C_1 = {64, f, 15, 14, n, walks, drinks, cow, normal},

C_2 = {64, m, 7, 5, s, lies, sleeps, heifer, normal}.

Now we received two tuples. One of them contains information similar to real cow features, the other one has false data. Later user can interpret the result and take further steps.

7. CONCLUSIONS

The integration algorithm we made proved to be reliable, provided the data structure sticks to the conditions we described. The complexity of the algorithm is very low so it can be used in virtually any environment requiring coming up with a consensus for a huge database. Any predefined organising of the data is not necessary, so the algorithm is complete in this form. The implementation proves that the algorithm returns correct results even for humongous datasets.

ACKNOWLEDGEMENTS

This paper was supported by Grant no. N N519 444939 funded by Polish Ministry of Science and Higher Education (2010–2013).

REFERENCES

[1] ARIELI O., AVRON A., *A model-theoretic approach for recovering consistent data from inconsistent knowledge bases*, Journal of Automating Reasoning, 2, 1999, 253–309.
[2] BADACHE N., HURFIN M., MADECO R., *Solving the consensus problem in a mobile environment*, [in:] Proc. of IEEE International Performance, Computing and Communications Conference, Piscataway, NJ, IEEE, 1999, 29–35.
[3] FRANKLIN S., GRAESSER A., *Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents*. Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag, 1996.
[4] HARANCZYK G., *Analiza skupień na przykładzie segmentacji nowotworów*, StatSoft Polska, 2005.
[5] LENZERINI M., *Data Integration: A Theoretical Perspective*, Madison, PODS, 2002, 233–246.
[6] LLOYD S. P., *Least square quantization in PCM*, IEEE Transactions on Information Theory, 28 (2), 1982, 129–137.
[7] NGUYEN T.N., *Advanced Methods for Inconsistent Knowledge Management*, Springer-Verlag, London, 2007, 47–70.
[8] NWANA H.S., *Software Agents: An Overview. Knowledge Engineering Review*, Vol. 11, No. 3, Cambridge University Press, 1996, 205–244.
[9] PANAIT L., LUKE S., *Cooperative Multi-Agent Learning: The State of the Art*, Autonomous Agents and Multi-Agent Systems, 11(3), 2005, 387–434.
[10] REIMER U., *Knowledge integration for building organizational memories*, [in:] Proc. of the 11th Banff Knowledge Acquisition for Knowledge Based System Workshop, Vol. 12, 1998, KM-61-KM-620.

network motifs, artificial networks,
social networks

Wojciech FRYS*
Krzysztof JUSZCZYSZYN*

GENERATION OF SCALE-FREE NETWORK GRAPHS WITH GIVEN MOTIF DISTRIBUTION

In recent years it was proved that calculating common graph properties, like average shortest path or clustering coefficient is too general for many complex networks which reveal non-random distribution of local topological structures – network motifs. In this paper we present algorithms to generate network graphs with two predefined properties, degree distribution and motifs distribution. The results may be used to fine-tune simulations modelling epidemic spread on networks, information cascades and other dynamic network phenomena.

* Wrocław University of Technology, Institute of Computer Science, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland, krzysztof.juszczyszyn@pwr.wroc.pl, Wojciech.frys@pwr.wroc.pl

PART 2

ARTIFICIAL INTELLIGENCE AND MULTIAGENT SYSTEMS

similarity function,
distance function, rough sets

Grzegorz POPEK*

SIMILARITY OF ROUGH SETS AND ITS APPLICATION TO KNOWLEDGE PROCESSING

Manual processing of vast amounts of knowledge is not only cost-inefficient, but becomes almost impossible assuming time restrictions. Distributed knowledge repositories crave for automated methods for documents processing. Usual descriptive complexity of documents can be reduced using approximate descriptions. Still, comparing documents described with rough sets remains an issue. We propose a common-sense approach for evaluating similarity (or distance) between documents described with rough sets. The task focuses on an interpretation of the border of a rough set and its influence on common knowledge processing tasks.

1. INTRODUCTION

1.1. BACKGROUND

Amounts of information nowadays can be no longer processed by one person or by a single machine. Distributed approaches to knowledge acquisition are relatively common, but at the same time they bring problems. Autonomy of sources of knowledge may result in inconsistencies in a body of knowledge as a whole. One of the key aspects of knowledge integration is the reduction of inconsistencies [7].

The task of knowledge integration [3, 7, 10] and its particular examples – integration of opinions, integration of observations, integration of demands – can be approached in a classical way by introducing a distance function into the knowledge domain and finding an optimal aggregate based on its distance from integrated knowledge items.

In general, the classic approach fails under some scenarios. The main suspect is usually the high level of inconsistency present in the body of knowledge. In such situations, trying to choose the middle ground between two groups of demands (or

* Institute of Informatics, Wroclaw UT, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław.

opinions) often renders the decision useless for both of them. However, as shown in [6], the traditional distance-based approach followed in this paper can be incorporated into multi-stage cluster-based knowledge integration frameworks (compare [1]) in order to deal with inconsistent bodies of knowledge.

1.2. SCENARIO

Let us assume the source of knowledge as a set of agents

$$Ag = \{a_1, a_2, ..., a_M\}. \tag{1}$$

Each agent is assumed to be equipped with mechanisms for interpretation and generation of Agent Communication Language (*ACL*) statements. They are fundamental for a provision of communicative functions within an agent and constitute a part of a communication interface. The core of the mechanisms consists of a set of procedures for an extraction of knowledge from obtained *ACL* messages and for a representation of the said knowledge using agent's internal representation.

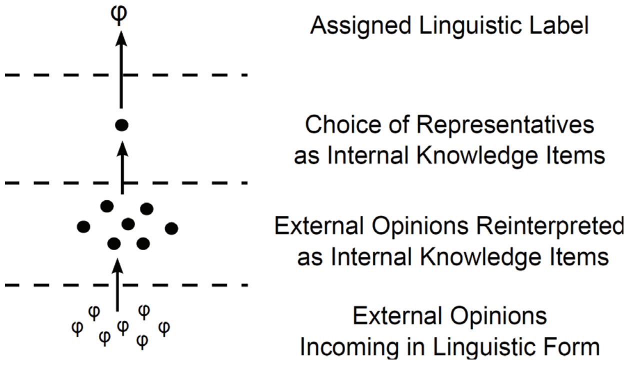

φ
Assigned Linguistic Label
Choice of Representatives as Internal Knowledge Items
External Opinions Reinterpreted as Internal Knowledge Items
φ φ φ φ φ φ φ
External Opinions Incoming in Linguistic Form

Fig. 1. Idealised Approach (see [6,9])

Agents' preferences (or demands) consist of objects $o \in O$. Knowledge about agents' preferences is provided to an integration agent a_0. It is transferred in form of incoming messages $\varphi \in ACL$ and needs to be re-interpreted, i.e. transformed into internal knowledge representation. As it can be seen in Figure 1, in the first stage of integration an agent a_0 receives external messages of the language ACL. For simplicity, we assume here (after [9]) that all messages (or opinions) relate to the same phenomenon (e.g. a particular property of a particular object) and need to be integrated.

2. PROBLEM DESCRIPTION

2.1. KNOWLEDGE REPRESENTATION

In this paper we assume, that the meaning of symbols used in an external language of communication can be represented using subsets of the global set of objects *O*. To avoid unnecessary complications related to a definition of ACL and its semantics let us directly assume, that

$$\varphi \in 2^O . \tag{2}$$

In contrast, knowledge at an internal level is represented using rough sets [8]. Rough sets are an important notion of approximate reasoning during decision making. The notion of a rough set is originally introduced using information systems.

Definition 1. *Information System*. Let a quadruple

$$S = \langle O, A, V, \rho \rangle . \tag{3}$$

be an information system, where

- $O = \{o_1, o_2, ..., o_S\}$ – a set of objects described in the system,
- $A = \{a_1, a_2, ..., a_K\}$ – a set of objects' attributes,
- $V = V_{a_1} \cup V_{a_2} \cup ... \cup V_{a_K}$ – sum of sets of values of particular attributes,
- $\rho : O \times A \to V$ is an information function.

Further, let an indiscernibility relation

$$\equiv \subseteq O \times O , \tag{4}$$

such that

$$\langle o_1, o_2 \rangle \in \equiv \quad \textit{iff} \quad \forall a \in A : \rho(o_1, a) = \rho(o_2, a) \tag{5}$$

be given. It is easy to see, that the relation ≡ is an equivalence relation, since it is reflexive, symmetric and transitive. In consequence, it generates a division Θ of *O* into equivalence classes.

Definition 2. *Division of O.* Let a set

$$\Theta = \{O_1, O_2, \ldots, O_M\} = \{[o]_{\equiv} : o \in O\} \tag{6}$$

be a division of O, where each O_i (for $i = 1, \ldots, M$) is non-empty (and different) subset of O and where $[o]_{\equiv}$ is an equivalence class of o, i.e., where

$$[o]_{\equiv} = \left\{o^* : \left\langle o, o^* \right\rangle \in \equiv\right\}. \tag{7}$$

It naturally follows, that sum of all O_i (for $i = 1, \ldots, M$) is equal to O.

Introduced notions can be further used in order to define an approximation of φ, that is, of a set of objects.

Definition 3. *Rough Set – Approximation of φ.* Each set of objects $\varphi \subseteq O$ can be described using the division Θ. The approximation is given as a pair

$$\varphi^{\equiv} = \left\langle \underline{\varphi}, \overline{\varphi} \right\rangle, \tag{8}$$

where

$$\underline{\varphi} = \{o : [o]_{\equiv} \subseteq \varphi\} \tag{9}$$

$$\overline{\varphi} = \{o : [o]_{\equiv} \cap \varphi \neq \varnothing\} \tag{10}$$

$\underline{\varphi}$ and $\overline{\varphi}$ are called a lower and upper approximation, respectively. Let us also denote a set of all potential approximations as

$$O^{\equiv} = \left\{\left\langle \underline{\varphi}, \overline{\varphi} \right\rangle : \varphi \subseteq O\right\}. \tag{11}$$

Rough sets are one of important notions used in decision making. However, the problem described in this paper is how to integrate multiple rough sets.

2.2. IDEALISED APPROACH TO KNOWLEDGE INTEGRATION

Messages obtained from agents from the set *Ag* are translated into an internal representation and form a knowledge profile represented as a multiset [5]:

$$\Pi = \{\varphi_i : i = 1, ..., M\}. \tag{12}$$

The theory of choice and consensus [7] makes it possible to define an idealised model for knowledge integration task [6, 9] for a knowledge profile Π. This task can be treated as equivalent to a problem of consensus choice [7].

Definition 4. *Idealised Knowledge Integration.* Let us assume a knowledge profile Π and similarity function [2]

$$\sigma : O^{\equiv} \times O^{\equiv} \to [0,1]. \tag{13}$$

The problem of consensus choice (or: a problem of idealised knowledge integration) is to find an element $\varphi^* \in O^{\equiv}$ such that

$$\varphi^* = \arg\max_{\varphi_i \in \Pi} \sum_{\varphi_j \in \Pi} \sigma(\varphi_i, \varphi_j). \tag{14}$$

A problem of consensus choice can be naturally described a minimization when a distance function [2] is used instead of similarity function.

As it has been already hinted in the introduction, an approach given by Definition 4 should be applied only if the knowledge profile is relatively consistent as such a situation guarantees that the consensus makes sense. Depending on an interpretation of the knowledge integration procedure, the approach can be used even for knowledge profiles with low consistency. However, in such situations it is advised to apply a multi-stage approach [6].

Finally, in situations in which there are multiple candidates satisfying formula (14), the consensus can be picked at random out of the candidates or some more elaborated approach can be applied (compare [9]).

2.3. COMPARISON OF ROUGH SETS

The main problem described in this paper is how to define a function present in Definition 4, that is, the similarity σ of two rough sets. While distance functions (metrics) are well-defined by a set of axioms, there is in general higher flexibility related to defining similarity functions. It should be noted, that [2] provides a set of axioms for similarity functions which is in a way equivalent to a set of axioms used for distance functions. Still, the majority of researchers accepts the following definition (and interpretation) of a similarity function.

Definition 5. *Similarity Function*. Let us assume a function

$$\sigma : O^{\equiv} \times O^{\equiv} \rightarrow [0,1]. \tag{15}$$

A function σ is a similarity function if and only if

$$\forall \varphi^{\equiv} \in O^{\equiv} : \sigma(\varphi^{\equiv}, \varphi^{\equiv}) = 1. \tag{16}$$

It is well-accepted that for two different elements a value of σ should be lower than 1. The value of 1 represents a maximal similarity. The value of function σ gets lower as compared elements get more different. It is also advised that the value of minimal similarity, that is – zero, is reached by the function for pairs of most different elements. Additionally, usually (16) is limited to hold only non-empty elements, which would translate in an assumed case to such $\varphi^{\equiv}$ which have non-empty upper approximation.

In some situations (16) may be weakened even further by limiting its application only to these "non-empty" rough sets, which have empty boundary [8], where boundary is understood as a difference between upper and lower approximation.

We will demand the similarity of $\varphi^{\equiv}$ with an empty upper approximation to any other rough set to be equal to zero, i.e.

$$\forall \langle \underline{\varphi}, \overline{\varphi} \rangle, \varphi^{\equiv} \in O^{\equiv} : \overline{\varphi} = \varnothing \Rightarrow \sigma(\langle \underline{\varphi}, \overline{\varphi} \rangle, \varphi^{\equiv}) = 0. \tag{17}$$

3. SIMILARITY OF ROUGH SETS

3.1. CLASSICAL APPROACH

The easiest approach to a design of similarity functions for rough sets' comparison is to ignore a fact, that they represent an approximation. One could ignore either lower or upper approximations and compare rough sets as if they were classical sets O_1 and O_2 using classical similarity functions [4]:

- Jaccard – $\dfrac{|O_1 \cap O_2|}{|O_1 \cup O_2|}$
- Dice – $\dfrac{2 \cdot |O_1 \cap O_2|}{|O_1| + |O_2|}$

- Overlap – $\dfrac{|O_1 \cap O_2|}{\min\{|O_1|,|O_2|\}}$
- Recall – $\dfrac{|O_2|}{|O_1 \cup O_2|}$
- Precision – $\dfrac{|O_1|}{|O_1 \cup O_2|}$
- etc.

It should be noted, that in some situations it may be important to pay attention to clusters defined by indiscernibility relation (i.e. to its equivalence classes) rather than to particular objects.

Definition 6. *Class-decomposition measure*. Let us introduce sets $O^* \subseteq O$, such that

$$\forall o \in O : [o]_{\equiv} \subseteq O^* \vee [o]_{\equiv} \cap O^* = \varnothing\,, \tag{18}$$

which can be naturally interpreted as sets which can be described as a sum of equivalence classes of a relation ≡. Such sets are assigned with a measure

$$\mu(O^*), \tag{19}$$

which equals to a number of equivalence classes of a relation ≡ from which O^* is composed.

It is easy to show, that if sets O_1 and O_2 are both either lower or upper approximations, then they both satisfy (19). Also, their sum $O_1 \cup O_2$ and their product $O_1 \cap O_2$ also satisfy (19). In consequence, previously proposed similarity functions can be adapted accordingly:

- Jaccard – $\dfrac{\mu(O_1 \cap O_2)}{\mu(O_1 \cup O_2)}$,
- Dice – $\dfrac{2 \cdot \mu(O_1 \cap O_2)}{\mu(O_1) + \mu(O_2)}$,

- Overlap – $\dfrac{\mu(O_1 \cap O_2)}{\min\{\mu(O_1), \mu(O_2)\}}$,
- Recall – $\dfrac{\mu(O_2)}{\mu(O_1 \cup O_2)}$,
- Precision – $\dfrac{\mu(O_1)}{\mu(O_1 \cup O_2)}$.

While in certain situations using such the classical approach may be advised, it will not be analysed here in length, as it is the most trivial one and an interpretation is deemed to be obvious.

3.1. NON-CLASSICAL APPROACHES

Usually, an interpretation of boundary regions [8] of rough sets should not be ignored. A non-empty boundary region (or shorter, boundary) is what makes a set non-crisp. Sometimes rough sets are modelled using an extension of a characteristic function of a classical by introducing a value 0.5. This value is assigned to these elements which belong to an upper approximation but do not belong to the lower approximation.

Elements of the boundary should have different influence on values of similarity function compared to elements of the lower approximation and compared to elements that are completely outside of the set (that is, to elements which do not belong to the upper approximation).

An influence of such elements can be weighted using a coefficient $\alpha \in (0, 1)$. Values of α which are closer to 1 represent situations in which elements of the boundary are treated with higher importance. Adequately, values of α which are closer to 0 assign elements of the boundary with lower importance.

In order to incorporate a coefficient α, Jaccard index can be extended in a following way[1]:

$$\sigma_{J1}(\varphi_1,\varphi_2)= \frac{\left|\underline{\varphi_1} \cap \underline{\varphi_2}\right|}{\left|\underline{\varphi_1} \cup \underline{\varphi_2}\right|} + \alpha \cdot \frac{\left|\underline{\varphi_1} \cap \left(\overline{\varphi_2} - \underline{\varphi_2}\right)\right|}{\left|\underline{\varphi_1} \cup \left(\overline{\varphi_2} - \underline{\varphi_2}\right)\right|} + \alpha \cdot \frac{\left|\left(\overline{\varphi_1} - \underline{\varphi_1}\right) \cap \underline{\varphi_2}\right|}{\left|\left(\overline{\varphi_1} - \underline{\varphi_1}\right) \cup \underline{\varphi_2}\right|}$$
$$+ \alpha^2 \cdot \frac{\left|\left(\overline{\varphi_1} - \underline{\varphi_1}\right) \cap \left(\overline{\varphi_2} - \underline{\varphi_2}\right)\right|}{\left|\left(\overline{\varphi_1} - \underline{\varphi_1}\right) \cup \left(\overline{\varphi_2} - \underline{\varphi_2}\right)\right|} \qquad (20)$$

[1] A symbol "–" is used for an operation of set-difference due to Equation 3.0 shortcomings.

Naturally, if again clusters defined by indiscernibility relation (i.e. to its equivalence classes) are more important than particular objects, one can reformulate the above to obtain a following similarity function

$$\begin{aligned}\sigma_{J1}(\varphi_1,\varphi_2) = {} & \frac{\mu\left(\underline{\varphi_1} \cap \underline{\varphi_2}\right)}{\mu\left(\underline{\varphi_1} \cup \underline{\varphi_2}\right)} + \alpha \cdot \frac{\mu\left(\underline{\varphi_1} \cap \left(\overline{\varphi_2} - \underline{\varphi_2}\right)\right)}{\mu\left(\underline{\varphi_1} \cup \left(\overline{\varphi_2} - \underline{\varphi_2}\right)\right)} + \alpha \cdot \frac{\mu\left(\left(\overline{\varphi_1} - \underline{\varphi_1}\right) \cap \underline{\varphi_2}\right)}{\mu\left(\left(\overline{\varphi_1} - \underline{\varphi_1}\right) \cup \underline{\varphi_2}\right)} + \\ & + \alpha^2 \cdot \frac{\mu\left(\left(\overline{\varphi_1} - \underline{\varphi_1}\right) \cap \left(\overline{\varphi_2} - \underline{\varphi_2}\right)\right)}{\mu\left(\left(\overline{\varphi_1} - \underline{\varphi_1}\right) \cup \left(\overline{\varphi_2} - \underline{\varphi_2}\right)\right)}.\end{aligned} \tag{20}$$

It can be seen that both similarity functions σ_{J1} and σ_{J2} – as mentioned in Definition 5 – do not reach the value of 1 when comparing a rough set to itself if a boundary of this set is non-empty.

Other classical similarity functions described in Section 3.1 can be transformed in the same way in order to take into account the boundary of rough sets. The main intuition conveyed within this transformation is that a non-empty boundary bears some uncertainty which should be influencing final values of similarity functions.

4. SUMMARY

Usual descriptive complexity of documents can be reduced using approximate descriptions. Still, comparing documents described with rough sets remains an issue. We have proposed a common-sense approach for evaluating similarity between documents described with rough sets. The approach focused on an interpretation of the border of a rough set.

The second proposed modification is to compose similarity functions using granulation of the domain imposed by an indiscernibility relation rather than using single objects. It would be possible to go one step further and define an influence of each cluster based on its size, but we leave this task for the reader due to the limited size of the paper.

ACKNOWLEDGEMENTS

This paper was supported by Grant no. N N519 444939 funded by Polish Ministry of Science and Higher Education (2010–2013).

REFERENCES

[1] CZARNOWSKI I., *Cluster-based instance selection for machine classification*, [in:] Knowledge and Information Systems, Springer-Verlag, Vol. 30(1), 2012, 113–133.

[2] DĄBROWSKI M., LAUS-MACZYŃSKA K., *Metody wyszukiwania i klasyfikacji informacji* (*Methods for searching and classification of information*), Wydawnictwa Naukowo-Techniczne, Warszawa 1978.

[3] HAN J., KAMBER M., *Data Mining: Concepts and Techniques*, 3rd ed., Morgan Kaufmann Publishers, Inc., 2011.

[4] KATARZYNIAK R., LORKIEWICZ W., POPEK G., *Relating documents and queries to ontological perspective of an electronic documents repository*, [in:] Ontologies and soft methods in knowledge management: Advanced Knowledge International, Adelaide, 2005, 45–72.

[5] LIPSKI W., MAREK W., *Analiza Kombinatoryczna* (*Combinatorial Analysis*), WNT, Warsaw 1986.

[6] LORKIEWICZ W., POPEK G., KATARZYNIAK R., *Intuitive approach to knowledge integration*, [in:] 6th International Conference on Human System Interaction, HSI 2013: Sopot, Poland, June 06–08, cop. 2013, IEEE, Proceedings, Piscataway, NJ, 2013, 1–8.

[7] NGUYEN N.T., *Advanced Methods for Inconsistent Knowledge Management*, Springer-Verlag, London 2007.

[8] PAWLAK Z., *Rough sets*, International Journal of Computer & Information Sciences, Vol. 11(5), October 1982, 341–356.

[9] POPEK G., KATARZYNIAK R., *Interval-Based Aggregation of Fuzzy-Linguistic Statements*, [in:] Proceedings of FSKD 2013, 23–25 July 2013, Shenyang, China, IEEE (to appear).

[10] SKORUPA G., LORKIEWICZ W., KATARZYNIAK R., *A Multi-agent Strategy for Integration of Imprecise Descriptions*, [in:] Proceedings of ACIIDS 2012, Vol. 1, 2012, 1–10.

naming game, language alignment,
collective knowledge

Wojciech LORKIEWICZ*

MEASURING THE COLLECTIVE ALIGNMENT OF INDIVIDUAL NAMING CONVENTIONS

In this paper we assume a multi-agent system involved in a collective alignment of naming conventions. Highly distributed and autonomous systems need to be endowed with the ability to establish and share utilised language structures. In particular, following a general model of the naming convention alignment process (language game model) we overview the behaviour of the multi-agent system. The major focus of this research is on the problem of how to describe an individual linguistic stance of an agent (agent-level) and relate it to a collective stance (system-level). Thus allowing for a quantitative research of the alignment process, both on agent- and system-level. In particular, we introduce and briefly discuss six distinct measures: success rate, language coherence rate, average number of used words, overall number of words, amount of synonymy and homonyms, strength of language associations and language strength.

1. INTRODUCTION

Language is an important tool that is extensively used in heterogeneous and distributed multi-agent systems to gain, share and utilise information in a social settings. It language allows individual agents to communicate with each other by exchanging meaningful messages. Moreover, linguistic communication facilitates the interplay between agents thus allowing to engage in social actions and even maintain social relations. As such systems are usually embedded in a hazardous and unknown environments it is crucial for a population of interacting agents to autonomously develop and control their behaviour, also in terms of linguistic communication. In particular, it is crucial that individual semantics (that form a shared and a coherent language) allow for maintaining consistent and understandable descriptions of the external environment despite its dynamic characteristics.

* Institute of Informatics, Wrocław University of Technology, Wyb. Wyspiańskiego 27, Wrocław, Poland.

The overall shape of a naming convention utilised by a collective results from two major forces – individual and population. The former controls the development of novel linguistic building blocks, understood as particular word-object associations. While, the latter tends modulate their usage to allow for a conventionalised definition of shared naming convention and thus allow for a successful communication within the collective. A particular word cannot function as a sign, thus serving as a particular example of denotation, until the audience (collective) distinguishes it. Moreover, it cannot successfully function in the collective until the audience agrees on its particular usage pattern. Following Berger (See [1]) "conversation/communication aims at reality-maintenance of the subjective reality. What seems to be a useless and unnecessary communication of redundant banalities is actually a constant mutual reconfirmation of each other's internal thoughts (maintains subjective reality)". In short, it's primary role is to maintain equilibrium in a social relationship.

The aforementioned dual nature of the communication directly implies a need for constant adaptation of the individual communication systems, and requires constant balancing between the two forces, i.e., thrives of the individual agents and trends in the collective. Consequently, as the individual agent is the only source of meaning of a linguistic symbol the collective language is shaped by individuals. In particular, it is only the individual agent that can introduce novel words and new word-object associations into the collective system. However, as the language goes beyond an individual agent each word-object association usage is modulated and shaped by the entire collective. Thus, the resultant meaning of linguistic symbols is a common agreement on a certain naming convention.

In this paper we overview a general model of the naming convention alignment process in multi-agent systems. In particular, we focus on a practical problem of how to describe the linguistic stance of an individual agent (agent-level) and how to relate it to the collective stance (system-level). After a short introduction of the process of semiosis in artificial systems (alignment process) we present a general description of the assumed multi-agent system, with a particular focus on the linguistic capabilities of an agent. Finally, we provide a detailed elaboration on the relation between the agent-level measures on the general, collective, stance of the system.

2. LANGUAGE ALIGNMENT AS A PROCESS OF SEMIOSIS

Language has its origin in an individual, as its basic blocks are shaped by individual agents. In particular, these individuals are further responsible for incorporating these novel blocks into the internal organisation of the individual systems. However, on the other hand, language is also shaped by the population, i.e., as its main and obvious function is to facilitate communication. In particular, these individuals are further responsible to the collective for a proper usage of these basic blocks, i.e., language structures.

There is a particular need for a shared naming convention that should be established between the interacting individuals. As an example, we must agree with the shop assistant that the real world object, that we intend to buy, relates to a particular name that we can mutually use. For instance, we need to share the idea that a particular word "apple" relates to the round green object lying in the crate. We can note that the semiotic relation between the sign and the object needs to be fixed beforehand (allowing communication).

Language structures need to be established and shared within the entire collective. In human societies such an institutionalisation of a used language relates to a fundamental process of shaping the basic language structures – process of semiosis. In essence, it is a dynamic process that happens within a group of communicating humans and characterises with the ability to shape and modulate a particular relation between a sign (form of external representation) and a particular embodied meaning (empirically perceived and grounded experience) within the group.

Following de Saussure (See [9]) "the meaning of a symbol in a given population results from a certain convention and is a result of a common agreement". Further, a group of individuals engaged in communication can be considered as a "dynamic" distributed cognitive system (See [10]), where the different participants collaborate in order to perform
a certain distributed, social and cognitive task. In particular, as noted by Cherry (See [3]) "words are signs which have significance by convention, and those people who do not adopt the conventions simply fail to communicate. They do not "get along" and a social force arises which encourages them to achieve the correct associations".

2.1. OBJECT LANGUAGE

In the cognitive linguistics (CL for short) approach a meaning relates to a form of mental representation (concept) derived from a particular precept[1] available in the external world. CL perfectly aligns with the presented approach, where a language symbol consist of a particular form-meaning pairing – the so-called symbolic assembly (See [4]). Following Russell (See [8]) a set of fundamental symbolic assemblies can be regarded as the object-language used by an individual.

The proper correlation between a particular word and a particular object (word-object association that can form a symbolic assembly) can only be learnt through associative learning. In essence, due to relatively more frequent correlation between presence of an object and uttered word in a communication act. At first, in terms of coherence of the collective – social denotation function of a sign – the association between

[1] For instance, a certain range of perceptual information derived from the surrounding world.

a word and an object is not likely to be correct. It is only incorrect, until the collected experience is filled with many, both positive and negative exemplars of a word usage. Further, having established the object-language an agent is able to determine a particular word for a given object, and vice-versa, a particular object for a given word. In practice, as soon as such an association is developed a word is considered to be "understood" (and "grounded").

In general, the sensorimotor capacity and in general cognitive abilities of an individual determine the space of all available associations, i.e., what can and what cannot be learnt by an agent. In this research we follow the Russell's object-language approach (See [8]) and the Sassurean approach that allow for an introduction of a quite substantial simplification of the general linguistic model. In particular, it justifies the assumed isolation of the perception mechanisms. As such we further omit details related to a particular implementation of agent's perception and assume that each individual is capable of successful determination of objects available in the environment and is capable of successful determination of linguistic utterances used by the collective. It should be noted that from a theoretical point of view given a sufficient capacity one can express in the object-language nearly every non-linguistic occurrence.

2.2. WORD LEARNING

In such a setting word learning is a rather simple task of mapping linguistic labels – words – onto a set of pre-established concepts – objects (See [2]). However, the problem is far more complex in a multi-agent setting. In particular, any differences in individual naming conventions can result in miscommunication, i.e., unsuccessful communication between interacting agents. As such, agents not only need to develop and sustain their individual naming preferences, but most importantly, need to additionally align them with the entire collective in order to form a coherent and shared naming convention (an object-language).

Each autonomous agent through a series of consecutive linguistic interactions with other agents is required to align its individual naming convention. It is certainly not a trivial task, as for instance an individual hearing a novel word, without any additional clue from the speaker, may presumable assign the meaning of the novel word to an infinite number of meanings. We can relate to a famous *gavagai* example from [11] and consider a linguist, studying a new language, that observes a native speaker exclaiming "gavagai!" in a situation where a small rabbit scurries by. The word "gavagai" could naturally mean the rabbit, but logically, it could mean an infinite number of many other different things, such as a certain part of the rabbit body, a running rabbit or even that it is a hot day. This is a well-known fact that is referred to as the indeterminacy of meaning. In short, the task of establishing the correct relations between labels and meanings in a population of communicating individuals is a notoriously hard problem.

2.2. LEXICON

In the assumed settings and from the linguistic point of view, the lexicon is the most important part of an agent. Following the object-wise perception and object-language the lexicon is understood as a collection of form-meaning couplings. It is typically represented as a form of an associative memory between meanings – objects - and forms – words. Such lexicons, like an embodied ontology, are strictly private and thus can differ among the agents. As aforementioned, a lexicon represents an internal association between words and objects. In practice, such a lexicon contains scored associations that indicate the strength of a word-object association, understood as a certain measure of the effectiveness of a particular association.

In essence, we follow the associationistic approach, where a lexicon L defines the strength $\delta^L(o, w) \in [0, 1]$ (correlation strength) of a particular mapping between an object $o \in O$ and a particular word $w \in W$. As such the lexicon encapsulates the current state of agent's language.

Speaking requires a "search" over the lexicon for an association that corresponds to the meaning of the selected topic (distinguished – in other mental processes – object from the external world) and identifies a particular word. Whilst, hearing requires a reverse "look-up", where the hearer searches for an association that corresponds to the uttered word and identifies a particular object. We can further, assess an act of communication in terms of its results. For instance, we indicate that a successful communication requires the speaker to using a linguistic clue (utterance) trigger a particular meaning (related to the intended meaning) 'within' the hearer.

In order to shape the naming convention utilised by an agent, the aforementioned word-object association scores, are further updated. Usually, after a single communication act its success is evaluated. If the act succeeds, the used association is reinforced (reinforcement), while competing associations are laterally inhibited (inhibition). If the act fails, the scores of the used associations are decreased. These basic rules of adaptation ensure that successfully used elements tend to be reused, while unsuccessful ones tend to die out. In that sense, a system comprised of multiple interacting agents with individual lexicons can be considered as a "complex adaptive system" (See [10]).

2.3. ALIGNEMENT PROCESS

In the literature there are several approaches proposed to the alignment of naming-conventions in multi-agent system. Nevertheless, the most relevant research was performed by Steels et al. (See [10]) introducing the Language Game Model (LGM for short) and formulating basic settings for the interaction between agents. In particular, the language game model defines a mechanism that through a simple interaction (and a simple learning mechanism) allows the agents develop a shared naming-convention. The basic game involves two agents – one speaker and one hearer – and a shared context (a given set of objects from the environment). During an interplay, both agents perceive the context. The speaker selects a single object (topic) and tries to produce (based on its lexicon) an utterance (word) to focus the attention of the hearer on the selected object. As a result, the hearer tries to correctly relate the linguistic utterance to a particular object in the current context. If the hearer succeeds, i.e., relates the linguistic clue to the intended topic, the game is considered a successful game – otherwise it's a failure.

The learning occurs through multiple co-occurrence between words and objects (cross-situational learning). In the simplest case, after each interaction the hearer updates its lexicon (reinforcing and inhibiting associations). In particular, it dampens the correlation between the received word and currently not perceived objects, and enforces the correlation between the received word and currently perceived objects, while the correlations with other words remain unchanged. Through a series of such interactions the agents are able to agree on a certain naming convention.

3. MULTI-AGENT SYSTEM

The model of semiosis is a complex model that consists of several sub models that define particular aspects governed within the process of resolution of the language symbols' meanings. In particular, we distinguished the model of the environment, the model of the agent, the model of population (and interaction), and the model of the lexicon (See [6]). As in this research we are focused solely on the relation between the individual and collective stance we do not provide detailed descriptions (See [5]).

3.1. AGENT

Due to the fact that each agent $a \in P$ from the population P of agents needs to be able to shape its language, its lexicon L^a must be flexible and adaptable. The value of correlation strength $\delta^{La}(o, w)$ between an object $o \in O$ and a particular word $w \in W$ changes over time, due to the interaction the agent is experiencing. In particular, the higher the strength is, the more definite the agent is that a certain word is an adequate name for a given object.

Each agent is able to interpret external utterance $\Gamma_I^a(w, L^a) \in O$, i.e., select the most adequate object (based on its current lexicon state), and produce external utterance $\Gamma_P^a(o, L^a) \in W$, i.e. as the most adequate name for a given object (based on its lexicon).

Consequently, the lexicon encapsulates the current state of agent's language, that for convenience can be viewed as a weighted complete bipartite graph $G = (V, E, \delta)$, where all words and objects are represented as vertices V, and all object-word pairs are represented as edges with a weight given by δ. As such, the actual graph structure of the lexicon modulates agent‘s interpretation and production scheme. In particular, both reflect a certain method of traversing the lexicon graph (a proper selection of edges according to the current distribution of weights).

4. MEASURING THE COLLECTIVE STANCE

In order to formulate the dynamics of the alignment process, we identify two major axes of elaboration, i.e., communication statistics (CS) and word statistics (WS). We focus on the evolution of the assumed multi-agent system and study the behaviour of the system based on six distinct measures: success rate (CS), language coherence rate (CS), average number of used words (WS), overall number of words (WS), amount of synonymy and homonymy (WS), the strength of language associations (WS) and the language strength (WS).

4.1. COMMUNICATION STATISTICS

The most straightforward measure is the frequency of successful communications (See Fig. 2) between agents. It resembles the observed ability of the system to transfer information from one agent to another, and as such it allows to reason about the utility of the communication system itself. In order to keep track of the effectiveness of the communication we calculate the success rate μ_{SR} of order N, as follows:

$$\mu_{SR}(N) = \left\langle \mathrm{I}[o_T == \Gamma_I^A(\Gamma_P^a(o_T, L^a), L^A)] \right\rangle_{t \in T|_N}, \tag{1}$$

where $<X>_Y$ is the average of X over Y, $T|_N$ is the set of time points of last N interactions, $I[.]$ is an identity function ($I[x == x] = 1$), Γ_I^A is agent’s A interpretation function, Γ_P^a is agent’s a production function, L^a is agent’s a lexicon and o_T is the topic.

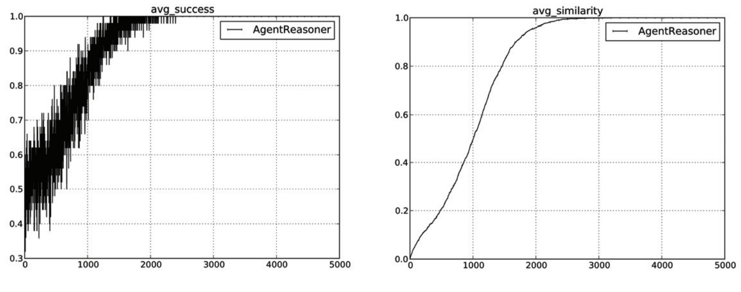


Fig. 2. Success rate (left) and coherence rate (right) dynamics during the language alignment process

In general, the success rate of order N is the frequency of successful communications in the last N interactions ($T|_N$). Despite its simplicity, this measure, in isolation, is not very useful. In particular, it does not take into account all objects from the environment, and can be easily deformed. For instance, agents communicating only about a single object are able to reach the highest possible success rates, as they might share a common name for a preferred object, despite having poor coherence between other names. Consequently, despite its simplicity the frequency of successful communications does not take into consideration the coherence between the entire space of names and focuses solely on objects involved in the communication.

Due to the restrictions of the success rate measure we formulate an additional measure that resembles the spread of a naming convention in a collective and reflects the coherence of names among all of the existing objects. We introduce language coherence μ_{CR} (See Fig. 2), as the probability that two randomly selected agents assign the same name for a randomly selected object from the environment, as follows:

$$\mu_{CR} = \left\langle I[o == \Gamma_I^A(\Gamma_P^a(o, L^a), L^A)] \right\rangle_{a,A \in P, a \neq A, o \in O}, \tag{2}$$

where O is the set of all available objects in the environment.

The lowest possible coherence, i.e. $\mu_{CR} = 0$ reflects a state of minimal language coherence in the system, as there are no two agents that use the same name for any of the objects. The highest possible coherence, i.e. $\mu_{CR} = 1$, represents the state of maximal coherence, where all agents share the same naming convention. It should be noted that in the assumed settings a system is absorbed by the coherent state, as from this point all of the utterances are consistent with the observed context, and without any external disturbance all of the strongest associations will remain the strongest.

4.2. WORD STATISICS

In order to analyse the characteristics of the emergent language we keep track of the number of used words μ_{UW}, the average number of invented words (per agent) μ_{IW} and the total number of all words μ_{TW}:

$$\begin{aligned} \mu_{UW}(\varphi) &= \left\langle \left\| \{w : \exists o \in O.\, \delta^a(w,o) > \varphi\} \right\| \right\rangle_{a \in P}, \\ \mu_{TW} &= \left\langle \left\| W^a \right\| \right\rangle_{a \in P}, \\ \mu_{IW} &= \left\langle \left\| W^a |_{INV} \right\| \right\rangle_{a \in P}, \end{aligned} \tag{3}$$

where $\|.\|$ is the cardinality function, W^a represents the set of words available to the agent a, and $W^a|_{INV}$ represents the set of words invented by the agent a.

The number of used words (See Fig.3) is calculated as the average, over all agents, number of positively associated words and it resembles the stability of current associations. In particular, the number of used words of magnitude $\varphi \in [0, 1]$ is the number of words that associated with a certain object with the value of association strength of at least φ. As the optimal communication system has one-to-one mappings between words and objects, i.e. the same number of used words as the number of existing objects, any deviation from this proportion reflects a potentially unstable situation, as miscommunication might occur. Obviously, for $\varphi = 0$ the number of used words relates to all positively associated words, whereas for $\varphi = 1$ it relates to only the words that have the maximal association. Consequently, the aforementioned magnitude φ determines the intended certainty, i.e., minimal certainty, of particular usage for a given word association.

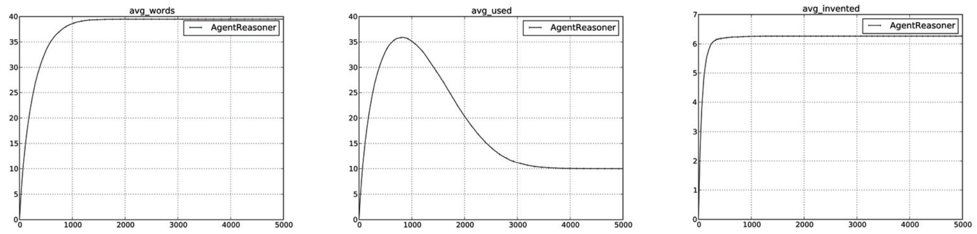


Fig. 3. The total number of words (left), the number of used words (middle) and the average number of invented words (right) dynamics during the language alignment process

Establishing the total number of words (See Fig.3) we need to calculate the overall number of existing words in the system. This measure resembles the stability of the communication system during its development phase. It should be stressed that new words may enter the lexicon, i.e. as agents are inventing new words, on a regular basis. However, as it is not possible for a word w to leave the lexicon (as the opposite mechanisms is a bit different) and a word w can become disassociated through the dampening procedure (weight of associations $\Sigma_o\delta(w, o)$ shared with a word w and/or usability $\gamma(w)$ is close to 0). Nevertheless, the higher the number of different words in the system, the significantly higher is the number of all possible associations and possibly lower coherence. Moreover the higher the number of words, both used and invented, the more technically demanding the system is. It needs more memory to store all of the associations and more of the processing power to cope with all possible association.

We also keep track of the "quality" of the current state of the language. In particular, as the idealised state involves one-to-one mapping between objects and symbols (such a setting allows for the highest success rate and maximal coherence) we further calculate the distance of the current language state to the one-to-one case, homonymy:

$$\begin{aligned} \mu_{HOM} &= \left\langle \|O\| - \left\| w : \exists o \in O. w = \Gamma_P^a(o, L^a) \right\| \right\rangle_{a \in P} \\ \mu_{SYN} &= \left\langle \left\| \{ w : \exists w \in W^a . \Gamma_I^a(w', L^a) = \Gamma_I^a(w, L^a) \,\|\} \right\| \right\rangle_{a \in P} \end{aligned} \tag{4}$$

In linguistic terms, the homonymy measure denotes the amount of homonymy present, on average, in the collective. In particular, it represents the average, among the entire population, number of words that a particular agent has associated with more than one object (in terms of production procedure Γ_P) and as such relate to the situation where multiple objects from the environment are named with a single word. Certainly, such a situation may cause inherent misunderstanding during communication.

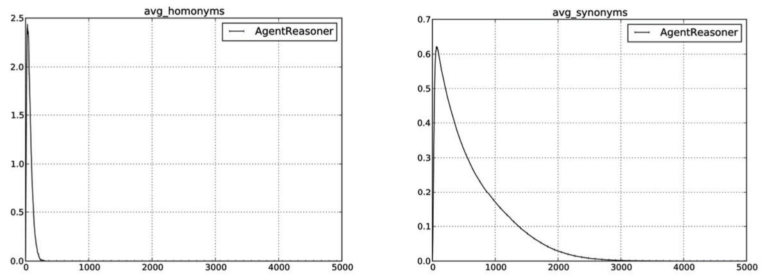


Fig. 4. The amount of homonymy (left) and synonymy (right) during the language alignment process

The second measure related to the "quality" of the current state of the language directly relates to the amount of synonymy (See Eq.4) present in individual lexicons. In particular, it represents the average, among the entire population, number of words that have the same object associated (in terms of production procedure Γ_P) with them. It directly relates to the situation where multiple of the used words relate to a single object. Consequently, such a situation causes the agents to use more words than required and might cause miscommunication in certain situations.

It should be noted that μ_{HOM} and μ_{SYN} take values from $[0, \|O\|]$ and $[0, \|W\|]$, respectively. In particular, $\mu_{HOM} = 0$ resembles lack of homonymy in the system, and $\mu_{HOM} = \|O\|$ resembles the maximal homonymy (all objects are named using a single word – a case of holistic language). Analogously, $\mu_{SYN} = 0$ resembles the lack of synonymy in the system and $\mu_{SYN} = \|W\|$ resembles the maximal synonymy (all words are associated with a single object).

Additionally, we can keep track of the alignment process from a point of view of associationistic approach. At first, we can measure the strength of associations that are incorporated in the used naming convention (used by an agent), i.e., word-object pairings that are actually used. In particular, a certain word-object (w, o) assembly is considered used by an agent if the agent uses the word w to name the object o. The language association measure reflects the stability of the used naming conventions, as high association values directly relate to a lexicon that requires a significantly more radical adjustment in order to introduce changes in the utilised naming conventions. Consequently, significantly more individual learning episodes are required to modify the currently established used language. At second, we can measure the strength of individual usability of words incorporated in the used naming convention (used by an agent). As the words are used by the agents to convey a particular (w, o) pair to other agents, each word can be associated with agent's subjective assessment of its usability $\gamma^a(w)$ that denotes agent's individual estimate of word's w popularity in the collective, i.e., strength of a particular word spread within the population. Consequently the language strength measure reflects agents individual assessment of the popularity of the used words. High utility values $\gamma^a(w)$ directly relate to a situation in which it is common for an agent a to register the word w in settings that are consistent with its interpretation of the word. Low values relate to a rather sporadic (or none) encounters of the word w, as the consistent settings (consistent with the interpretation of the word) are more commonly associated with utterances of other words.

$$\begin{aligned}\mu_{LA}(\varphi) &= \left\langle \delta^a(o,w) : \exists o \in O. \Gamma_P^a(o,L^a) = w \wedge \delta^a(o,w) > \varphi \right\rangle_{a \in P} \\ \mu_{LS} &= \left\langle \gamma^a(w) : \exists o \in O. \Gamma_P^a(o,L^a) = w \right\rangle_{a \in P}\end{aligned} \tag{6}$$

where $\gamma^a(w)$ represents agent's individual notion of usability of a particular word w.

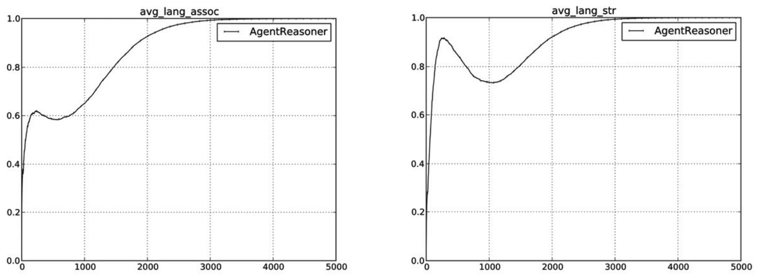


Fig. 5. Language association (left) and language strength (right) during the language alignment process

It should be noted that both, $\mu_{LA}(\varphi)$ and μ_{LS}, take values from the unit interval [0, 1]. In particular, for a given magnitude $\varphi \in [0, 1]$ the value of $\mu_{LA}(\varphi)$ resembles the current strength of the associations forming the agent's language. As such, $\mu_{LA}(\varphi) = 0$ resembles a naming convention that is relatively (depending on the magnitude φ) weak, whereas $\mu_{LA}(\varphi) = 1$ resembles a naming convention that is relatively (depending on the magnitude φ) strong. Similarly, the value of μ_{LS} resembles the current strength of the usability $\gamma^a(.)$ of words used in the language. As such, $\mu_{LS} = 0$

resembles a situation, where the used naming convention is comprised of unpopular (in the population) words, whereas $\mu_{LS} = 1$ resembles a naming convention that is comprised of highly popular words in the collective. The higher the value of the aforementioned measures, the better is the quality and the stability of the utilised naming convention.

5. CONCLUSIONS

Language has its origin in an individual, as its basic blocks are modified and developed by individual agents. These individual systems relate linguistic structures to their internal organisation. However, as its main function is to facilitate communication within the collective a language is mainly shaped by the population.

In this paper we have overviewed a general model of the naming convention alignment process in a multi-agent system from the quantitative perspective. We have identified two major axes of elaboration, i.e., communication statistics and word statistics. We gave a brief description of the behaviour of the system in terms of six distinct measures: success rate, language coherence rate, average number of used words, overall number of words, amount of synonymy and homonymy, strength of language associations and language strength.

The introduced rich set of measures on the one hand allows for a detailed look at the individual system, and on the other enables a holistic view on the alignment of individual naming conventions. Moreover the proposed measures allow for a practical and comprehensive approach towards studying the dynamic behaviour of the alignment process, for instance in case of different settings, different alignment strategies, etc.

ACKNOWLEDGEMENTS

This paper was supported by grant no. N N519 444939 funded by polish ministry of science and higher education (2010–2013).

REFERENCES

[1] BERGER P., LUCKMANN T., *The Social Construction of Reality: A Treatise in the Sociology of Knowledge*, Penguin Books, 1966.
[2] BLOOM P., *Mindreading, communication and the learning of names for things*, Mind & Language, Vol. 17(1–2), 2002, 37–54.
[3] CHERRY C., *On human communication; a review, a survey, and a criticism*, The Technology Press of MIT, Oxford, 1957.
[4] EVANS V., GREEN M., *Cognitive linguistics: An introduction*, Edinburgh University Press, 2006.
[5] LORKIEWICZ W., KATARZYNIAK R., *Issues on Aligning the Meaning of Symbols in Multiagent Systems*, New Challenges in Computational Collective Intelligence, Studies in Computational Intel-

ligence, Vol. 244, 2009, 217–229.
[6] LORKIEWICZ W., KATARZYNIAK R., KOWALCZYK R., *Individual Semiosis in Multi-agent Systems*, Transactions on Computational Collective Intelligence, Vol. 7190(VI), 2012.
[7] PIERCE C., *Collected Papers of C. S. Peirce*, Harvard University Press, Cambridge, 1935.
[8] RUSSELL B., *An inquiry into meaning and truth*, WW Norton & Co., 1940.
[9] SAUSSURE F.D. *Course in General Linguistics*, Open Court Pub Co., 1986.
[10] STEELS L.A., *Language as a complex adaptive system*, Parallel Problem Solving from Nature PPSN, Vol. VI, 2000, 17–26.
[11] QUINE W.V.O., *Word and object*, The MIT Press, 1960.

*description logic, ontology, OWL, rule-based systems,
knowledge representation, ITSM, ITIL, decision support*

Adam CZARNECKI, Tomasz SITEK*

ONTOLOGIES VS. RULES – COMPARISON OF METHODS OF KNOWLEDGE REPRESENTATION BASED ON THE EXAMPLE OF IT SERVICES MANAGEMENT

This text provides a brief overview of selected structures aimed at knowledge representation in the form of ontologies based on description logic and aims at comparing them with their counterparts based on the rule-based approach. Due to the limitations on the length of the article, only elements associated with the representation of concepts could be shown, without including roles. The formalisms of the OWL language were used to record ontologies, while the rules were expressed in Prolog.

To better illustrate these two ways of knowledge representation, examples of best practices from the field of IT services management were used, which are contained in a set of publications known as the *Information Technology Infrastructure Library* (ITIL).

The purpose of the comparison is to examine the possibility of using an ontological approach in situations where the use of rule-based solutions is problematic.

1. INTRODUCTION

Information technology (IT) tools for knowledge representation (KR) are used for acquiring new knowledge through inference, namely knowledge which is not declared explicitly, on the basis of knowledge and information already gathered.

The concept of knowledge representation categorizes the cognitive and creative values of all intellectual human activities and behaviours. Representation is connected with the realization of systems development processes and affects the effectiveness of project solutions. It is also important that knowledge is both the source and the result of IT processes carried out in decision making [1, p. 14]. There are two basic types of symbolic knowledge representation:

* Gdańsk University of Technology, Faculty of Management and Economics, Department of Applied Informatics, ul. Narutowicza 11/12, 80-233 Gdańsk, Poland.

- procedural representation — aimed at identifying a set of procedures, the operation of which is represented by the knowledge of the field,
- declarative representation — aimed at identifying a set of facts, statements and rules specific for a particular domain, (with limited information on how to use them).

The advantage of procedural representation is the high efficiency in describing processes (e.g. recognized laws). Whereas declarative representation is easier in the area of description and formalization. The authors' study is devoted to this type of representation. Among the KR methods, two relatively common ones can be pointed out: (1) rule-based knowledge bases, (2) ontologies.

In this paper, the authors' objective is to compare the two methods of knowledge representation in the form of a case study. The area which will constitute the basis for this discussion will be best practices in Information Technology Service Management, (ITSM) called *Information Technology Infrastructure Library* (ITIL) Version 3. The study will present some selected structures used in the creation of ontologies from the above-mentioned area, accompanied by their rule-based representations (or attempts at representation). Thus, ontologies will be the reference point, as they are a more complicated method, but one which gives more possibilities. The rule-based approach will be expected, therefore, to allow the modelling of the same knowledge on the basis of its characteristic expressiveness.

The language used in the ontology description is OWL 2, as the implementation of the $\mathcal{SROIQ(D)}$ dialect. The rule-based representation relied on Prolog.

2. KNOWLEDGE REPRESENTATION METHODS

2.1. RULES

Processing knowledge and building mechanisms which perform automatic inference have been done for years. The first attempts to use formalized knowledge for decision-making were made in the 1950s and 60s. Work in this area then was focused on the development of general principles of intelligent problem-solving (works by Newell, Simon and their successors). GPS – General Problem Solver created by Newell and Simon could actually solve certain logical tasks with its built-in algorithm seeking the way in the space of available states [2].

Although it showed its successors the direction to go, this concept proved to be insufficient [3]. The approach, which turned out to be reasonable, was one in which the processing of knowledge from a particular field required primarily extensive domain knowledge. The logical machine did not have to be complex [4, p. 31]. The key to the effective use of knowledge by inference mechanisms lay in the choice of an appropriate method of knowledge representation.

One of the most important methods of declarative knowledge representation in decision support systems are facts and/or rules as well as ontologies, which are briefly characterized below.

Facts (or statements) refer to such issues as events, phenomena, symptoms, activities. Statements are usually stored in the form of O-A-V:

(<OBJECT> , <ATTRIBUTE> , <VALUE>) (1)

Usually, a set of statements (facts) is not sufficient to describe a domain in a complete manner. Knowledge bases, which include rules in addition to statements, are the basis for most expert systems created so far. In general, a rule is presented as follows:

IF <premise> THEN <conclusion> (2)

Specific methods (languages) of programming, called symbolic languages, were developed especially for intelligent technologies. In principle, symbolic languages were to facilitate the development of intelligent systems, as they offered procedures designed to encode knowledge, build inference mechanisms and conduct operations using symbolic expressions made up of, for example, phrases, rules or facts. [5, p. 43]. More sophisticated methods encompass so-called association rules [6].

One of the most important languages for expressing phrases, rules and facts is Prolog (from French: *Programmation en Logique*). It was created in the 1970s, in order to allow the mapping of complex logical relationships. Prolog made it possible to perform the so-called logical programming, more specifically in the language of Horn clauses (implications with only one conclusion). The concept of Prolog is based on predicate logic, and proving theorems involves a resolute system of denials. Prolog can be distinguished from classic programming languages due to its possibility of interpreting the code in both a procedural and declarative way [7]. Because the lines of code are a direct record of logical relationships, they can be read as declarations of certain relationships between sentences.

Starting a program written in Prolog results in the execution of a deduction process, namely drawing conclusions from premises identified as input to a logical problem. Thus, the task of the creator of the algorithm boils down to defining "what the problem is" and not – as is the case for procedural languages – "how to solve the problem" [8]. There are many implementations of this language, which vary in terms of the runtime environment, attached libraries or even the code editor. The free SWI-Prolog is one of the most common implementations [9]. It has also been used by the authors for their research.

2.2. ONTOLOGIES

The term "ontology" was borrowed by information technology in the mid-1960s from philosophy — for it is the name of a discipline which focuses on the study of the essence of being. What is widely understood by this term is a set of well-defined terms (vocabulary) regarding a specific field (domain) accepted by the community associated with that field [10, p. 267]. The word "model" does not appear in the definition, nonetheless, it can be assumed that this is what ontology actually is for the field – a model describing concepts and relationships between them, or simply a knowledge representation model.

The definition may not be sufficient to answer the question why it is worth creating ontologies. At least a few reasons can be pointed to [11, p. 84]. Ontologies are created:

- to spread a common understanding of the structure of information among people or agent-based applications,
- to enable the reuse of domain knowledge,
- to openly define assumptions regarding a chosen field,
- to separate the knowledge of the field from the knowledge related to operating that field,
- to analyze the knowledge of a particular field.

In practice, there are ontologies distinguished by varying degrees of formality – from a predefined vocabulary to knowledge models based on logic [12, pp. 26–28], especially description logic (DL). The latter constitutes a basis for the OWL (Web Ontology Language), its DL species (version 1), and a newer version – OWL 2 [13]. It is a language with high expressiveness, which carries the risk of creating an ontology for which conducting a correct inference in a finite time will be impossible. OWL 2 was used in the research presented in this publication. Apart from the aforementioned expressiveness, the choice was supported by the fact that this language is considered to be the recommended (and therefore standard) one by the World Wide Web Consortium (W3C) for Semantic Web technology and by the availability of tools for ontology modelling based on OWL (the application Protégé 4.X).

In the context of this paper, it should also be mentioned that there is an extension of OWL for a rule-based system – *Semantic Web Rule Language* (SWRL) [14]. However, it does not substitute expressions characteristic for ontologies, but adds the possibility of taking advantage of the rules which are in accordance with Horn clauses. Therefore, its existence does not refute the main idea of this paper – the indication (as much as possible) of rule-based substitutes for ontological statements.

3. THE DOMAIN

The knowledge contained in publications describing a set of best practices in IT service management, ITSM, called *Information Technology Infrastructure Library* (ITIL) Version 3 was selected to illustrate the examples comparing the use of ontological and rule-based constructions. The ITIL publications include 5 books, each of which is devoted to one of the phases constituting the complete cycle:

- IT Service Strategy [15],
- Service Design [16],
- Service Transition [17],
- Service Operation [18],
- Continual Service Improvement [19].

ITIL is at the top of the list of standards used by IT support organizations to support management functions, providing a set of guidelines on the provision of IT services, and at the same time, achieving a higher level of maturity and performance.

The scope of concepts used in ITIL is wide and its presentation would go beyond the narrow framework of this publication. The choice of an ITIL field to demonstrate the expressions of the OWL language and the rule-based approach stems from the authors' interest in researching this area, as was reflected in earlier publications [20–22].

4. CONCEPTS

In terms of ontology, concepts are formal representations of notions from the real world. Implemented in OWL they are called classes. These classes can be understood as sets which may be general categories of more detailed classes (then a hierarchy of concepts is created), or can contain concrete individuals.

To represent the knowledge of concepts and the relationships between them, the following – often together – can be used: logical expressions such as subsumption, equivalence, disjointness and negation, which are presented below. It should be noted, however, that these examples do not exhaust the range of structures available to a knowledge engineer. Due to limited space, expressions using roles and quantifiers have not been included.

4.1. THE HIERARCHY OF CONCEPTS

Creating the hierarchy of concepts is one of the most common applications of ontology. It is based on the use of a subsumption which is defined in description logic in the following way:

$$B \subseteq A, \tag{3}$$

where concept A subsumes concept B (Fig. 1).
A subsumption has a transitive nature and thus a classic syllogism takes place:

$$B \subseteq A \wedge C \subseteq B \Rightarrow C \subseteq A \tag{4}$$

In OWL, this structure is realised as the following expression:

SubClassOf(B A) (5)

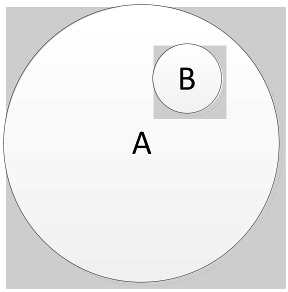


Fig. 1. Subsumption of concept B by concept A

This means that concept B is contained in concept A. A practical application of the hierarchy of concepts in the ITIL ontology will be an indication that the Service Desk belongs to the set of ITIL functions:

SubClassOf(Service_Desk ITIL_Function) (6)

whereas, every ITIL function is an asset according to this standard, which is written as:

SubClassOf(ITIL_Function ITIL_Asset) (7)

According to equation (4), the following can be concluded from equations (6) and (7):

SubClassOf(Service_Desk ITIL_Asset) (8)

Representing the same knowledge with the use of rules requires writing down two facts for each hierarchy and one general rule.

```
subClassOf(iTILFunction,serviceDesk).                          (9)
subClassOf(iTILAsset,iTILFunction).                            (10)
subClassOf(X,Z) :- subClassOf(X,Y), subClassOf(Y,Z).           (11)
```

In this case, the rule works as a recursion and allows the inference engine of Prolog to obtain information about all the possible hierarchical relationships, even these with multiple nesting.

4.2. EQUIVALENCE OF CONCEPTS

The equivalence, or identicalness of concepts appears when individuals which are instances of these concepts are always identical [1, p. 115]. This is written as follows:

$$B \equiv A \quad (12)$$

In OWL, the same sense is presented by the following structure:

EquivalentClasses(B A) (13)

For the ITIL field assumed in this paper, the following phrase:

EquivalentClasses(ITIL_Core_Book ITIL_Phase) (14)

means that the concept ITIL_Core_Book representing one of the five books on the essential phases of IT service management is equivalent to the concept ITIL_Phase, namely a phase of ITIL.

In this case, the Prolog language will mainly need a rule speaking about commutation, which is characteristic to equivalence:

equivalentClasses(X,Y) :- equivalentClasses(Y,X). (15)

Then all such dependencies between concepts will need to be determined in the form of facts. An example of a fact will have the following form:

equivalentClasses(iTIL_Core_Book, iTIL_Phase). (16)

It should be noted that it has been assumed here that the equivalence is always a relationship between two concepts. For more identical elements, the above example would have to be more complex (for three variables, there would already be six rules pointing to commutation).

4.3. DISJOINTNESS OF CONCEPTS

Disjointness of two concepts means that there can be no individual which would simultaneously belong to both of them [1, p. 116]. Thus, the intersection of the two disjoint concepts is an empty concept (⊥), which can be written as:

$$A \cap B \subseteq \bot \quad (17)$$

In OWL, the disjointness of concepts A and B is written as follows:

DisjointClasses(A B) (18)

The enumeration of more than two disjoint concepts should be interpreted as the disjointness of each pair of concepts.

Declaring the disjointness of concepts is one of the most frequently used structures in creating ontologies based on description logic. This results from the Open World Assumption, OWA, in which two concepts are not considered to be different from each other as long as it is not expressed directly or it cannot be inferred from other evidence.

An example of applying the disjointness of concepts in ITIL ontology is presented below:

DisjointClasses(ITIL_Application ITIL_People) (19)

Phrase (19) means that the two concepts which are specific instances of the ITIL_Asset concept (not mentioned here) – a specific set of concepts for ITIL assets, and referring to applications (ITIL_Application) and people (ITIL_People) involved in IT service management – are disjoint.

As for the rule-based record of disjointness, it is based on the same assumptions as that regarding their identity. Thus, there will be one general rule and respectively as many facts as arise from the observed cases of disjointness:

disjointClasses(X,Y) :- disjointClasses (Y,X). (20)

disjointClasses(iTIL_Application,iTIL_People). (21)

4.4. THE NEGATION OF THE CONCEPT

Sometimes the easiest way to say what a thing is, is to point out what it is not. This is one of the possible applications of the structure called the negation of a concept. To write that concept B is not concept A, we will use the following notation:

$$B \subseteq \neg A \tag{22}$$

While transferring negation onto the level of OWL ontology and language, concepts should be regarded as classes which are specific sets of potential individuals. Then the statement “not A” means “anything but A”. Thus, the following identities are true:

$$\top \equiv A \cup B \tag{23}$$

$$\bot \equiv A \cap B \tag{24}$$

They mean that A and B – disjointly — fill the whole universe $\top$. This is shown on Fig. 2.

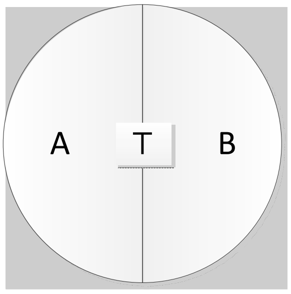


Fig. 2. Disjointness of concepts A and B expresses negation and encompasses all the universe $\top$

In OWL, the notion (22) will look as follows:

ClassAssertion(ObjectComplementOf (A) B) (25)

In reference to the ITIL ontology, it is possible to present knowledge about the fact that an organization which does not follow this set of best practices (NonITILOrganization) is the opposite of an organization in which the ITIL elements have been implemented (ITILOrganization). It is important to narrow down the conceptual scope to only the concepts subsumed by the idea of an organization, which can be achieved by applying the intersection of the concepts (their common part). Thus, the herein described knowledge can be expressed in the following manner:

EquivalentClasses(

NonITILOrganization ObjectIntersectionOf((26)

Organization ObjectComplementOf(ITILOrganization)))

How should this be done in the declarative knowledge base of the Prolog language? The essence of negation must be expressed with a rule using the predicate ‘not’.

organizationType(iTILOrganization, organization(X))

:- (27)

not(organizationType(nonITILOrganization, organization(X))).

To apply this rule, one would need to define any newly added organization also in terms of whether it belongs to the group of entities possessing ITIL or not. The following two facts allow the conclusion that Gdańsk University of Technology (GUT) does not belong to this group.

organization(gUT) (28)

organizationType(noniTILOrganization,organization(gUT)). (29)

5. FORMS OF KNOWLEDGE REPRESENTATION

Structures used in this text are listed in Table 1. It can be noticed that rule-based counterparts of ontological forms have successfully been found. Nonetheless, with the exception of negation, each of the rule-based examples requires more statements to express the same knowledge about concepts.

In addition, the authors studied the structures using roles and quantifiers: specific and general. However, due to space constraints, they were unable to include this aspect in this publication. For the ontological representations of such structures, satisfactory rule-based forms have not been found.

Thus, it seems reasonable to say that for the field of ITIL used in the research, based primarily on the relationships between concepts, the approach which uses ontologies gives greater possibilities of modelling the field when less complex structures of a chosen language are used.

Table 1. List of ontological and rule-based knowledge representations

Knowledge type	DL	Ontology (OWL)	Rules (Prolog)
Hierarchy of concepts	$C \subseteq B \subseteq A$	SubClassOf(B A) SubClassOf(C B)	subClassOf(X,Z) :- subClassOf(X,Y), subClassOf(Y,Z)
Equivalence of concepts	$B \equiv A$	EquivalentClasses(B A)	equivalentClasses(X,Y) :- equivalentClasses(Y,X)
Disjointness of concepts	$B \cap A \subseteq \bot$	DisjointClasses(B A)	disjointClasses(X,Y) :- disjointClasses (Y,X)
Negation of concepts	$B \subseteq \neg A$	ClassAssertion (ObjectComplementOf(A) B)	classAssertion(Y, X) :- not(classAssertion(Z, X))

5. CONCLUSION

The text provides an overview and a rough comparison of ontological and rule-based knowledge representation methods for the chosen typical issues of modelling concepts and their relationships. The aim of this comparison was to search for answers about the validity of applying ontology rather than rules for the chosen field of a set of best practices of ITIL.

The presentation of results was limited to basic equations describing concepts in the field. Nonetheless, even at this level of generality, the prevalence of the approach based on ontologies and the OWL language over the pair "rules–Prolog" was revealed. This stems from a richer semantic base present in the elements of the OWL language, which results from basing it on at least two lower language layers – Resource Description Framework (RDF) and RDF Schema (respectively: meta-meta-language and meta-language for OWL). In such an approach, Prolog remains rather at the level of meta-meta-language.

For a more complete picture of the comparison, other important structures would be needed, such as the roles combining concepts and the constraints applied along with them, such as "exists", "all", "at least", "at most", "exactly". Moreover, in the presented examples, some hypothetical statements about ontological representation being more useful have been successfully illustrated, but no formal proof was attempted. These issues will be addressed by the authors in their further research and publications.

REFERENCES

[1] BOLC L., BORODZIEWICZ W., WÓJCIK M., *Podstawy przetwarzania informacji niepewnej i niepełnej*, Warszawa, Państwowe Wydawnictwo Naukowe, 1991.

[2] NEWELL A., *A guide to the general problem-solver program GPS-2-2*, Rand Corp., Santa Monica, California, 1963.

[3] NEWELL A., SHAW J.C., SIMON H.A., *Report on a general problem-solving program*. Proceedings of the International Conference on Information Processing, UNESCO House, Vol. 2, No. 5222, 1959, pp. 256–264.

[4] NIEDERLIŃSKI A., *Regułowo-modelowe systemy ekspertowe rmse*, Wydawnictwo Pracowni Komputerowej Jacka Skalmierskiego, Gliwice, 2006.

[5] MULAWKA J.J., *Systemy ekspertowe*, Wydawnictwa Naukowo-Techniczne, Warszawa, 1997.

[6] WEICHBROTH P., *Odkrywanie reguł asocjacyjnych z transakcyjnych baz danych*, Prace Naukowe Uniwersytetu Ekonomicznego we Wrocławiu, A. Nowicki, I. Chomiak-Orsa (eds.), No. 82. Informatyka ekonomiczna. Rynek usług informatycznych, No. 14, Wydawnictwo Uniwersytetu Ekonomicznego we Wrocławiu, Wrocław, 2009, pp. 301–309.

[7] SITEK T., *Ocena języków systemów ekspertowych dla celu implementacji baz wiedzy Systemu Wieloagentowego*, [in:] Zarządzanie technologiami informatycznymi: przykłady zastosowań IT, C. Orłowski (ed.), Pomorskie Wydawnictwo Naukowo-Techniczne, Gdańsk, 2007, pp. 102–112.

[8] SITEK T., *Technologie informatyczne wykorzystywane w projektowaniu i implementacji systemów inteligentnych*, [in:] Zarządzanie technologiami informatycznymi: stan i perspektywy rozwoju, C. Orłowski (ed.), Pomorskie Wydawnictwo Naukowo-Techniczne, Gdańsk, 2006, pp. 69–82.

[9] *SWI-Prolog's home*, http://www.swi-prolog.org. [access: 2013-08-10].

[10] AUFAURE M., LE GRAND B., SOTO M., BENNACER N., *Metadata and Ontology-Based Semantic Web Mining*, [in:] Web Semantics Ontology, Idea Group Publishing, Hershey, London, Melbourne, Singapore, 2006.

[11] CZARNECKI A., *Technologie informatyczne wykorzystywane w projektowaniu i implementacji ontologii*, In: Zarządzanie technologiami informatycznymi: stan i perspektywy rozwoju, C. Orłowski (ed.), Pomorskie Wydawnictwo Naukowo-Techniczne, Gdańsk, 2006, pp. 83–94.

[12] GOCZYŁA K., *Ontologie w systemach informatycznych*, Akademicka Oficyna Wydawnicza EXIT, Warszawa, 2011.

[13] HITZLER P., PARSIA B., PATEL-SCHNEIDER P.F., RUDOLPH S., *OWL 2 Web Ontology Language Primer*, October 2009.

[14] HORROCKS I., PATEL-SCHNEIDER P.F., BOLEY H., TABET S., GROSOF B., DEAN M., *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*, http://www.w3.org/Submission/SWRL/ [access: 2013-08-10].

[15] IQBAL M., NIEVES M., *Service Strategy*, Office of Government Commerce, London, 2007.

[16] LLOYD V., RUDD C., *Service Design*, Office of Government Commerce, London, 2007.

[17] LACY S., MACFARLANE I., *Service Transition*, Office of Government Commerce, London, 2007.

[18] CANNON D., WHEELDON D., *Service Operation*, Office of Government Commerce, London, 2007.

[19] CASE G., SPALDING G., *Continual Service Improvement*, Office of Government Commerce, London, 2007.

[20] CZARNECKI A., ORŁOWSKI C., *Application of Ontology in the ITIL Domain*, [in:] Information Systems Architecture and Technology: Service Oriented Networked Systems, A. Grzech, L. Borzemski, J. Świątek, Z. Wilimowska (eds.), Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, 2011, pp. 99–108.

[21] PASTUSZAK J., CZARNECKI A., ORŁOWSKI C., *Ontologically Aided Rule Model for the Implementation of ITIL Processes*, In: Advances in Knowledge-Based and Intelligent Information and Engineering Systems, M. Graña, C. Toro, J. Posada, R.J. Howlett, L.C. Jain (eds.), IOS Press, 2012, pp. 1428–1438.

[22] PASTUSZAK J., CZARNECKI A., ORŁOWSKI C., *Ontology-Driven Rule-Based Model for an Extension of Information Technology Infrastructure Library Processes*, Cybernetics and Systems, Vol. 44, No. 2–3, 2013, pp. 245–263.

*full-text databases, bibliographical databases,
distributed databases, web and cloud computing*

Grzegorz BLINOWSKI*, Henryk RYBIŃSKI*,
Tomasz RAMSZA**, Tomasz KUSTRA**

ALPHA-ISIS – WEB- AND CLOUD-ORIENTED INFORMATION STORAGE AND RETRIEVAL ENVIRONMENT

In this paper we present α-ISIS – a ground-up reimplementation of the micro-CDS-ISSI database system. The CDS-ISIS database is a popular software system, used for generalized information storage and retrieval. It has been maintained by UNESCO since 1985. It was designed mainly for bibliographical applications. At the time of its implementation its notable features were advanced text searching capabilities, complex thesauri, and multi-linguality. The objective of our work was to open the system to new technologies (including novel storage databases, UNICODE standard, XML data representation), and make α-ISIS suitable for large databases. Most notable changes were done in the backend: it was redesigned to use a full-text Lucene open source system [4], as the indexing/search core, SQL (SQLite, PostgreSQL), and noSQL (MongoDB) backends can be used for information storage. A set of new features has been added on the functional level, mainly – remote document indexing, multi-database search, dynamic meta-indexes, web services support and cloud support via MongoDB.

1. BASIC CONCEPTS AND FEATURES OF THE ISIS SYSTEM

It is important to distinguish the "ISIS platform" from a plethora of different ISIS-related applications, software systems, tools and APIs. Broadly taken, the "ISIS platform" should be considered as a certain "philosophy" of building a bibliographical system on the foundation of data representation, data indexing and data formatting concepts. The API, physical data representation and the actual implementation has evolved heavily since the early versions were introduced. Let us describe briefly the three above mentioned concepts:

* Institute of Computer Science, Warsaw University of Technology, ul. Nowowiejska 15/19, 00-665 Warszawa.
** CC Otwarte Systemy Komputerowe Sp. z o.o., Rakowiecka 36, 02-532 Warszawa.

ISIS systems are intended to be used for structured non-numerical databases containing mainly texts, and are specialised in handling variable-length information. An ISIS "**Master File**" (**MST**) – is a logical database, which in general is storage-independent. Interestingly enough, the idea of a heterogeneous data structure came far before the nowadays XML and no-SQL database. The ISIS master may store various records of variable length, each being of another "type" and/or having another structure (by means of the set of attributes). For example in a bibliographic database one can store records describing books, journals, papers from journals, chapters from books, each type having different attributes. The records can be distinguished in various ways, for example by a specially selected field that keeps type of record. An ISIS record consists of fields of different types and formats, furthermore a field may be repeatable (like authors in a paper). Some fields may consist of subfields, which can be individually extracted and processed. ISIS is a schema-less approach. From the database point of view there are no particular requirements on the records' field composition, and little (or none in the new ISIS) requirements on size.

The MST database may be populated from various sources: MARC files* legacy text files, flat text files with fixed width columns, Excel spreadsheets, XML files (e.g. according to particular schemata such as Dublin Core [9]), BiBTeX, etc. Also, it is quite typical for an ISIS system to be able to serialize output data to multiple formats.

Inverted file is a logical name that refers to the index file, which provides a mechanism for fast record search and retrieval. Inverted file data is commonly generated from MST via two processes: inverted file generation (reindex operation) or inverted file update. ISIS SDK provides all the functions for processing and maintenance of the inverted files (reindex, update, findterm, getnext, etc). For searching with the inverted file two basic modes are provided by the ISIS API: search and browse. Both relay on the indexing mechanism. The role played by indices in ISIS is of much higher importance than for a typical RDBS system – they not only ensure required performance, but provide thesaurus assisted search, and handle complex queries – where multiple fields or an expression over multiple fields is used as a one search term. In ISIS there is and **FST** definition file (Field Select Table), which provides information on how to build index from the ISIS records stored in MST. FST is a crucial element of the ISIS concept – it contains all the specifications that are needed to build the index for a given master file.

The last ISIS-specific element is the formatting language (**PFT**) which is used to define precise formatting requirements for database records. The formatting language provides commands for complex handling of texts contained in ISIS records by means of (repeatable) fields and subfields. With PFT constructs one or more specific data

* MARC (MAchine Readable Catalogue format) is a format used by US Library of Congress for storing blbliografic data.

elements may be selected, sorted and optionally merged with literals, e.g. to label some or all the fields. PFT is a Pascal-like language. A collection of formatting commands is called a **format**. In general, a format defines a way of building texts from the ISIS record (possibly connected to other ISIS records). Such a text, generated from a record, can be then used in various ways, *inter alia*:

- It can be used for displaying warning and/or errors, when built by a validation format;
- It can be used for displaying the record on the user screen (for proofreading, for presenting search results, etc.);
- It can be used for building a search key for sorting records;
- It can also be used for building an export record (in a specific XML, e.g. Dublin Core format, or MARC XML);
- It can be used for providing a text to the index builder (in this case the text is built by a format from FST).

The formatting language is therefore the core of ISIS operations.

2. A BRIEF HISTORY OF THE ISIS SYSTEM

ISIS database is a popular software system used for generalized information storage and retrieval. Its beginnings date to the late 60s and the mainframe era, the original version being developed by Mr. Giampaolo Del Bigio for UNESCO's Computerized Documentation System (CDS) – it was based on the ISIS system (Integrated Set of Information Systems) at the International Labor Organization in Geneva. The wide proliferation was gained by the MS-DOS version – CDS/ISIS (Computerized Documentation Service / ISIS) [2] developed by UNESCO in 1985. CDS/ISIS was written in Pascal. WINISIS – the Windows version, was first demonstrated in 1995, it could run on a single computer or in a local area network.

The first efforts to put ISIS systems on the Web started in 1995 and resulted in "WWW-ISIS 1.0" developed jointly by ICIE and Warsaw University of Technology Computer Science Institute. WWW-ISIS was based on the original MS-DOS executable run under Linux (a wrapper library was used), a set of shell scripts was written to interface the executable to the front-end, the latter being implemented as a CGI application run under Apache Web Server. This approach, although functionally effective, lacked both in stability and performance, hence a more complex reengineering was required.

Since 2000 many solutions were provided to address the problem of porting the original MS-DOS/Pascal versions to a Web environment. Among them a leading role was played by BIREME (Biblioteca Virtual em Saúde), where ISIS-DLL [1] has been developed. ISIS-DLL is an "application programmer interface" (API) to the ISIS databases, packaged as a standalone MS-Windows dynamic-link library. With the tools provided by BIREME it became feasible to develop middleware for advanced web-based systems. Two development lines have started in more or less the same time: one at BIREME, resulted with WXIS project, and another one at ICIE, as a result of the FAO-WAICENT request, at the preliminary stage for building tools for AGRIS centers, leading to WWW-ISIS 2.0 or WWW-ISISPL [8].

A number of sophisticated systems were developed on the basis of ISIS-DLL, however after some 10 years since its introduction, it became clear that the ISIS-DLL as a MS-Windows API based software has some serious limitations making it practically unfeasible in new projects: for example – ISIS databases, as they were, were not Unicode compliant. This disadvantage drastically restricted applications of WWW-ISIS for multilingual databases, as well as multi-alphabet documents, which becomes quite restrictive in the text applications. Yet another problem was that ISIS-DLL based systems could be installed only on the Windows platform. An upgrade of the BIREME ISIS-DLL was considered, but after careful source code analysis it became clear that a complete re-write is a better option. Instead of changing the BIREME's ISIS-DLL which is strongly restricted by some concepts, nowadays a bit obsolete (like maximum record length, fixed and old structure of b-trees, with fixed length of indexes), we have implemented the new α-ISIS software completely from the scratch.

3. USERS AND APPLICATIONS OF ISIS SYSTEMS

Since the release of CDS/ISIS in 1985 over 20,000 licenses have been issued by UNESCO and a worldwide network of distributors, the primary uses being the catalogues of many small and medium-sized libraries. A number of different projects were started as a "spin-off" from the CDS/ISIS version, and later from the ISIS-DLL library. Below we would like to focus on the projects that led to the creation of the α-ISIS system.

As a result of cooperation between FAO (Food and Agriculture Organization of the United Nations), ICIE and CC* a whole line of software systems based on WWW-ISIS^{PL} have been developed since 2000. Created originally as a tool for capacity building and for distributing FAO information resources on CD-ROM (AGRIS and FAOBIB [3]), it very quickly turned out to be useful enough for larger applications at FAO and other institutions and libraries. The development of the FAOBIB cataloguing system and its OPAC (Online public access catalog for the

* Both, ICIE and CC are small private Polish companies, being kick-off's from Warsaw University of Technology.

FAO Lubin Library) was the driving force behind the development of new WWW-ISIS^{PL} functionalities. In 2002 on the request of FAO-WAICENT (World Agricultural Information Center Portal) WWW-ISIS^{PL} was subject of further development by ICIE and CC. As an outcome of this development, and following the successful co-operation between FAO, IFAD (International Fund for Agricultural Development) and GTZ (Deutsche Gesellschaft für Internationale Zusammenarbeit) in the area of information and knowledge management, the three organizations have decided to work on a common development project of the loan and acquisition module for their libraries.

With the experience gained, while building and adopting the library applications FAO and then: ICCROM (International Center for the Study of the Preservation and Restoration of Cultural Property), WFP (United Nations World Food Programme) and IDLO (International Development Law Organization), decided to build the library integrated system, which has been named WebLIS [13]. Fortunately, UNESCO also recognized that the idea of releasing WebLIS would become very important for the entire ISIS community, and as a result, UNESCO has decided to support its publication and distribution on their web site. In 2006 FAO has decided on integrating WEBLIS with WEBAgris, which resulted in a common FAO, CATIE (Centro Agronómico Tropical de Investigación y Enseñanza) and ICIE project, giving rise to a successful library integrated system known as LISAGR – currently the most advanced WWW-ISIS^{PL} based system consisting of multi-language: Cataloguing, OPAC, LOAN, Acquisition and Statistical modules.

4. α-ISIS ARCHITECTURE

The idea of reengineering the old ISIS system came up with the growing problems of running larger and larger ISIS databases, as well as, with various requests of making the applications closer to modern nowadays information retrieval systems. In particular the Faolex* application has reached a point, where no further development was feasible. So, the FAO Legal Office has decided to start a project with Warsaw University of Technology on a new ISIS.

As previously stated, the major goal of the reengineering that led to the development of α-ISIS was to upgrade information retrieval by adding full text retrieval, support large datasets (i.e. of the size of Gigabytes and larger), improve performance, and ensure the compatibility with such technologies like Web- and cloud-oriented computing. In effect, a software which can be described most broadly as "*ISIS API with ad-*

* Faolex is one of the most important ISIS-based FAO's application. It is a prominent world-wide legal database.

ministration tools" was created. The scope of reengineering was quite vast – for example a previously used Web fronted (WWW-ISIS) based on the ISIS-DLL had to be adapted to the new API, however on the front-side most of the compatibility was sustained, also from the data input and output perspective (batch import/ export formats as well as formatting constructs (PFTs) could be adapted from the earlier systems. It was not possible to ensure a 100% compatibility with the old API for the following two major reasons: larger data set handling required the change of some datatypes, Unicode support enforced some fundamental changes in the way that strings are handled.

An overview of the α-ISIS architecture is presented on Fig. 2. The α-ISIS API is a dynamic library written mostly in C/C++ – two version are available: a DLL for Windows environments and SO (shareable object) for Linux. The library is linked directly by the user interfaces. Three interfaces are currently available: a Web front-end compatible with most of Linux and Windows Web servers and a Windows/Linux GUI client (α-GUI), a command-line tool (α-Tool) is also available for the batch tasks, such as import/export, backup, restore, full-reindex, etc. The web front-end used by us was WWW-ISIS^{PL}, adapted to the new α-ISIS API. With moderate changes most of the Web application designed for ISIS-DLL could be used. The web application is a compiled C/C++ program. The CGI standard [12] is used on the web server. Due to the fact that a new process is created for each HTTP request, the CGI technique is considered to be less efficient than other solutions, but in our experience it suffices in the real-world applications – even for high-traffic sites, such as the FAOLEX system. The CGI interface has one important advantage over the non-CGI solutions, such as e.g. NSAPI, ISAPI, JavaEE, etc. – it is namely a light-weight and portable solution. For example – in one of the WWW-ISIS projects a light-weight WWW server was used to service local requests for a catalogue run directly from a CD-ROM. The other advantage of using the CGI interface became apparent during the many year life span of WWW-ISIS^{PL} development – the CGI standard is technologically stable since 1995, this let us maintain a steady evolution of the WWW-ISSI^{PL} software with focus on the user-level features, rather than following the changes of the Web server interface technologies.

The API itself interfaces with an SQL backend (PostgreSQL, or SQLite), both used as a storage for the ISIS records. Also MongoDB [10] may be used for the same purpose – with this backend it is possible to distribute the MST storage across multiple network nodes. The Lucene library [4] is used to handle indexing. Lucene – currently the "state-of-the-art" full-text database engine was chosen at the early stage of the project due to its performance and flexibility. The C/C++ Lucene interface is used in α-ISIS. External documents may be downloaded via HTTP from a remote database running an ISIS-database gateway – as shown in the "remote dB" part of figure.

The OBJ subsystem of the API is responsible for obtaining and serializing external documents. The OBJ module can extract, process and cache local and remote documents' meta-data. The OBJ module uses the Tika Open Source library, written in Java.

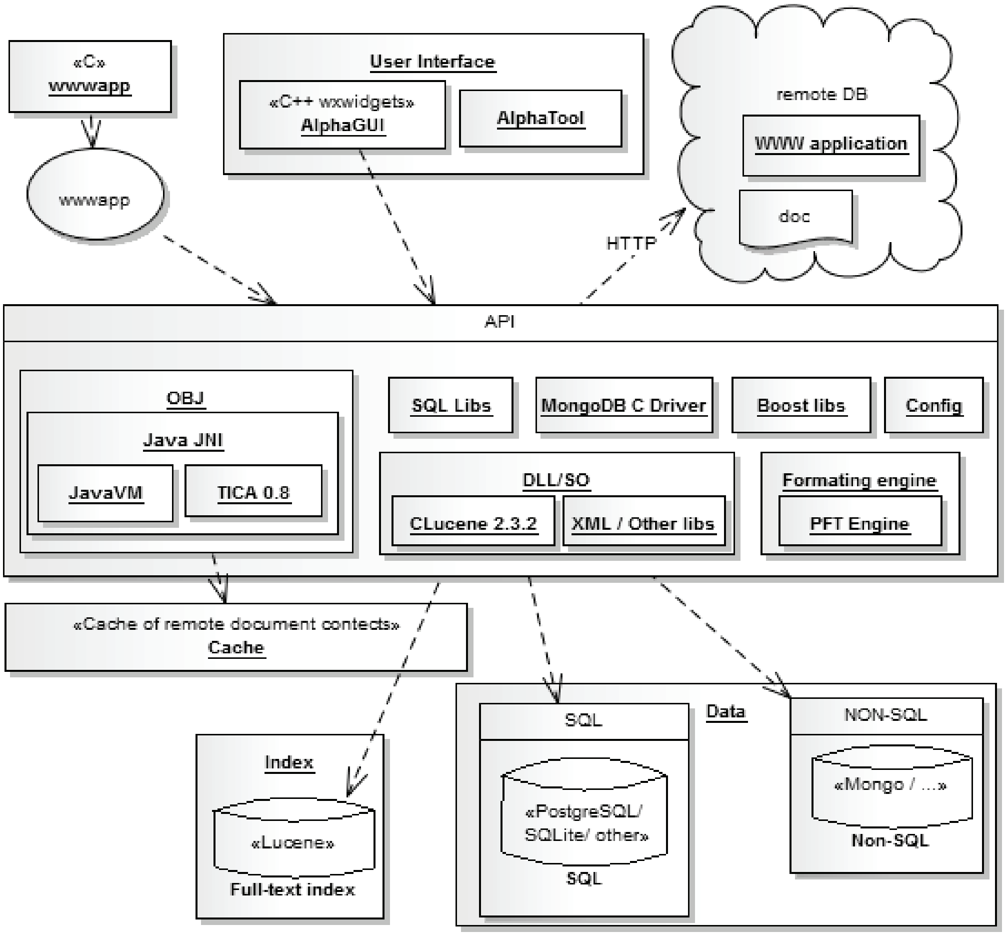


Fig. 2. An overall architecture of α-ISIS API and its environment

5. INDEXING AND META-INDICES (FST)

An index (inverted file) plays a central role in the ISIS system. α-ISIS indexes implemented by Lucene backend are always full text, but data extracted from records may be pre-processed to create different index types. Depending on the index type non-ASCII and/or special characters, and/or whitespaces may be removed and indexed text may be converted to lowercase.

Consider an example record shown in table 1 (to simplify the example we omit index types altogether).

Table 1. Sample MST record, "Keyword" and "descr" are repeatable fields, PublisherInfo has three subfields defined as "a", "b" and "c"

Record field name	Record field value
RecID	LEX-FAOC002623
Date	20020126
Title	Water Regulations
Author	John Doe
CCode	ZIM
PubInfo	^aHarare^bMinistry of Agriculture^c1966
Keyword	114
Keyword	034
Descr	Mathematics
Descr	Physics

A set of sample index definitions (FST) for such data scheme is shown in table 2.

Table 2. Sample indices for data shown in Table 1

Nr	Index name	Index definition	Description
1	ID	**v**RecID	This indexes field "1". v command extracts value from field
2	KEY_1	**v**Keyword [1]	The [] operator extracts n'th occurrence of a repeatable field Keyword
3	KEY_NUM	(**v**Keyword/)	The () Operator will return all occurrences of a given field merged and separated by '/' - newline
4	PI_YR	***v***PubInfo^c	The ^ operator extracts named subfield
5	TERRIT	**if** ***a***(vTerritory) **then** 'N' **fi**	The a() function returns True if the record contains no occurrence of the field or subfield indicated by the argument, here a "binary" index is build
6	A_MATH	**if** **v**Descr [1]=' Mathematics' **then** **v**Author) **fi**	If first occurrence of "Descr" field matches given string the value of "Author" field is used to build an index
7	CNTR_ORG	if ***nocc***(***v***CCode) <> 1 **then** 'ZZ' **else** ***ref*** ('valid', CC:" ***v***CCode";***v***Country) fi, ***v***PubInfo^a	nocc() function returns number of occurrences of a given field. *ref*() function extracts data from a different database, first parameter is database name, second - the query (here we search for records with 'CC matching country code, the third parameter is the output format – here simply country (name) will be returned.
8	TDYN	**if** ***v***Ccode = 'EUR' **then** (|EUR_D:|' ***v***Descr/) **else** (|OTHER_D:| ***v***Descr/) **fi**	The repeatable literal enclosed by the "|" sign defines text to be output only if the associated field or subfield is present in the record. If the field is repeatable, the literal will be repeated for *each* occurrence of the field. The colon expression *index:keyword* will add all descriptors from field Descr to index EUR_D or OTHER_D.
9	ALL	***v****	v* returns contents of all fields this index will be therfore a full textindex of the whole record

In the α-ISIS formatting language the expressions in FST are used to generate indices. In general PFT syntax is quite complex – the constructs like: conditionals, arithmetical and logical expressions are allowed. A set of ISIS specific functions which operate on fields, subfields, substrings, and perform pattern matching inside a record are also available [6]. Consider examples from Table 2:

- In examples 1,2,3, 4 and 9 a text index is built from: a data field, repeatable field, a subfield of enumerated repeatable field and the whole record respectively.
- Example 5 shows a simple if ... then construct: a binary index "TERRIT" is updated for this record if the field is not present.
- In example 6 the contents of "author" field is added to index A_MATH but only if the record's first descriptor is 'Mathematics'.
- Example 7 shows the use of external database reference – if Country code field is present index will be updated by a string merge of Country (name) taken from the external database "valid" and subfield "a" from PubInfo field, for the sample record "Zimbabwe Harare" will be returned.
- Example 8 shows a dynamic index – the index name 'TDYN' is only a placeholder – depending on the Ccode field match either EUR_D or OTHER_D index will be updated (if a given index is not present in the database it will be created).

A special *ref*() function has been incorporated to the formatting language in order to link records from various databases, also from remote systems. The syntax of the function is as follows:

ref(database;query_expression;format[;sort_format;lucene_sort_indx; from; to])

All the parameters except *from* and *to* are provided as PFT expressions. The ref() operation is somehow similar to the SQL-join, however the links between the records are based on Lucene querying, so that linking "similar" records is much more powerful than can be achieved in SQL by the *like* predicate. Additionally, it allows an arbitrary formatting of results returned by a referenced database, and the formatting is done by the referenced database (sometimes remote). Calls to the ref() functions may be nested. It is also possible to refer to an external database which is accessed via Web interface (remote PFT). In such case the referenced record is remotely formatted and returned to the originating database. This feature allows to create a distributed network of cross-referenced ISIS databases. Also recursive scripts can be run for formatting taxonomies, thesauri, and other hierarchical structures.

The last example in table 2 shows a creation of dynamic index: the name of the index is generated on the fly by the formatting expression. In our simple example the index name is given as a static string, but it is possible to generate index names from field contents. Hence, it is for example possible to generate a separate index for each author or country. This is a powerful feature because it allows to create hundreds of indices with just few statements. In practical applications it is used to generate separate keywords indices for each language.

6. HARVESTING MULTIPLE DOCUMENT FORMATS (OBJ)

The “OBJ” subsystem of the API is responsible for obtaining and serializing external documents. The syntax of the obj() function is: *obj(pft_object_data_url, pft),* example uses are:

- *obj('c:\Users\Tomasz\Downloads\MongoDB-Manual-master.pdf'; V**) – returns the whole serialized record, which is built from a local file,
- *obj('http://docs.mongodb.org/master/MongoDB-manual-master.pdf'; Vo_creator)* – extracts the o_creator field from the serialized remote site accessed via HTTP,
- *obj(|http://faolex1.fao.org/faolex/docs/|D650, if v650:'.DOC' then |texts/|V650, else if v650:'.PDF'then|pdf/|V650, else if v650:'.HTM'then|html/|V650,,fi,fi,fi,; „V**) – the object’s URL is formatted with PFT according to the contents of the field 650;

The module uses the Apache Tika toolkit [13] written in Java. It detects and extracts metadata and structured text content from various documents, including HTML, XML, Microsoft Office, ODF, PDF, ePUB, RTF, plain text, mbox, and others. Tika is also able to extract meta-data from the binary formats, such as audio and video. To avoid unnecessary serialization OBJ uses a cache of downloaded and serialized documents, which is used every time a partial or full reindex is required. If the document is not cached and must be serialized, it is: downloaded, a JVM environment is initiated and IsisTikaWraper is executed to invoke to Tika Java library. Tika returns a document in XHTML which we convert to XML, finally XML tags are processed as record fields by the formatting engine. Some of the meta-data that is extracted from the document includes title, author, document name, creation date and last modification, content-length, revision, keyword, etc. The document content can also be used and indexed by Lucene.

7. CONCLUSION

We have briefly described features of the α-ISIS system - an effect of over 15 years of software evolution which begun with a MS-DOS based system and currently supports all major Linux and Windows environments, uses advanced backend features while still servicing the growing needs of the wide user community. The system is still under development, the major planned extensions to the current version include: new backends for MST storage, LDAP and OpenId integration for user handling, and XSLT record formatting.

REFERENCES

[1] BIREME, *ISIS Application Program Interface – ISIS_DLL User’s Manua*, São Paulo, July 2001, http://bvsmodelo.bvsalud.org/download/isisdll/ISIS_DLL-Manual.pdf

[2] CDS/ISIS database software, http://portal.unesco.org/ci/en/ev.php-URL_ID= 2071&URL_DO= DO_TOPIC&URL_SECTION=201.html
[3] FAO, *AGRIS (International System for Agricultural Science and Technology)*, site: ttp://agris.fao.org/
[4] HATCHER E., GOSPODNETIC O., *Lucene*, Manning Publications Co., 2004.
[5] KALOYANOVA S. et al., *Standards for Agricultural Information Resources Management: AGRIS Application Profile,AGROVOC and LISAGR*, In: XIV Reunión Interamericana de Bibliotecarios, Documentalistas y Especialistas en Información Agrícola (RIBDA) Conference, November 6–10, 2006, Mexico City, Mexico, ftp://ftp.fao.org/docrep/fao/010/ag870e/ag870e00.pdf
[6] RYBIŃSKI H., KUSTRA T., RAMSZA T., *α-ISIS formatting language*, In: Institute of Computer Science (ICS) Warsaw University of Technology, Research Report 5/2011, 2011.
[7] RYBIŃSKI H., KUSTRA T., RAMSZA T., *α-ISIS Application Program Interface, ISIS SDK User's Manual*, In: Institute of Computer Science (ICS) Warsaw University of Technology, Research Report, 2010 *Preliminary Version*, In: Institute of Computer Science (ICS) Warsaw University of Technology, Research Report, 6, 2011.
[8] RYBIŃSKI H., BLINOWSKI G., RAMSZA T., α-WWW/ISIS Technical Reference Manual v. 1.0, In: Institute of Computer Science (ICS) Warsaw University of Technology Research Report 6/2011.
[9] KUNZE J., BAKER T., *The Dublin Core Metadata Element Set*, In: RFC5013, August 2007.
[10] MongoDB.org, MongoDB Reference, specification site: http://docs.mongodb.org/manual/reference/
[11] TIKA, Apache TIKA – a content analysis toolkit, http://tika.apache.org/
[12] W3C, *CGI: Common Gateway Interface v. 1.2*, specification site: http://www.w3.org/CGI/
[13] WEBLIS – CDS/ISIS database software, UNESCO and Information processing tools; http://portal.unesco.org/ci/en/ev.php-URL_ID=16841&URL_DO=DO_TOPIC&URL_SECTION=201.html

*multiagent system, semi-hierarchical partition,
knowledge integration, consensus theory*

Grzegorz POPEK, Michał ADAMSKI,
Łukasz BURDKA*

A MULTIAGENT SYSTEM FOR CONSENSUS-BASED INTEGRATION OF SEMI-HIERARCHICAL PARTITIONS

An outline of a multiagent platform for knowledge integration is presented. The input knowledge is obtained from a distributed group of agents in a form of semi-hierarchical partitions. Information about the world is gathered by observer-agents and later sent to central agent to form a knowledge profile. The central agent integrates the knowledge profile using methods adapted from consensus theory in order to choose a fitting representative. Central agent deals with inconsistency of information. First, the agent rates a consistency of the knowledge profile and if the consistency level is not satisfying, a clustering process initiates. Once the knowledge profile becomes divided into clusters, a separate representative is evaluated for each of its consistent parts. A method for choosing optimal number of clusters is presented. A prototype of the platform has been developed in order to provide empirical results.

1. INTRODUCTION

In distributed multiagent systems it is essential to integrate knowledge possessed by particular units. Unfortunately, in many cases the knowledge body obtained as a simple sum of knowledge of separate units is inconsistent [4]. Knowledge integration is especially important in distributed multi-agent systems where agents try to work out a single opinion about shared environment. Different methods of coping with this issue have been already studied and developed [1, 3, 4, 6]. In this paper a method of integrating knowledge represented by semi-hierarchical partitions is presented. Each partition represents an opinion of a particular agent on a state of the environment. Environment consists of objects, which are being categorized by agents. The opinion is often subjective, incomplete and/or contains errors. System is designed in

---

* Institute of Informatics, Wroclaw UT, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław.

such a way that through knowledge integration methods it tries to reduce the inconsistency in opinions and strives to create a common view on the environment.

Despite decentralized nature of the system, a particular agent is chosen as a representative performing the integration (compare [3,6]). She asks selected observer-agents for their opinion on the environment. In next step she finds the image of the environment that generates minimal distance to every opinion delivered by observers. If the result is unsatisfactory, clustering process is performed and – as a result – one or more consensus candidates are generated, which – in consequence – increases an overall consistency. Presented strategy of choosing consensus has relatively low complexity and works fast even when coping with large number of environment objects. Efficiency has been improved by choosing low-complex evaluation method (from those presented in [4]) and simple clustering algorithm. Implemented solution has polynomial complexity in contrast to commonly presented NP-complete solutions.

In this paper a prototype of the multi-agent system that integrates knowledge is presented. In Section 2 knowledge integration task is described. Section 3 is focused on finding consensus method. Section 4 presents methods for coping with inconsistency. In Section 5 a prototype of a framework is presented. Section 6 contains results obtained from the system.

2. KNOWLEDGE INTEGRATION TASK

There is a demand for knowledge integration in many multiagent systems. Opinions about the same environment developed by several agents may vary. In order to work out consensus, which is an output opinion mostly fitting input from all agents, there is a need to define distance function between two opinions in order to determine how similar they are. Then evaluation function for the consensus has to be selected. Our system is designed to deal with inconsistent data while categorizing certain object from selected environment.

A single opinion developed by one agent is a tree consisting of classifications of environment objects, which the observer was able to classify. Semi-hierarchical partitions received from each agent and forming a profile may be incomplete or inaccurate. Due to this issue knowledge possessed by central agent is inconsistent and cannot be presented as result. Thus it has to be integrated. Overall goal of integration task is to develop result, which is an optimal opinion generated from possessed knowledge.

In this case optimal solution is the one which generates minimal distance to each input partition. It's chosen from all possible environment states. Output has the same form as a single input – semi-hierarchical partition. It is expected to be as close to actual environment state as possible.

2.1. FINDING CONSENSUS

When several experts present their opinions about certain subject there is a need of finding consensus. One opinion mostly accepted by agents. Our system takes advantage of consensus theory presented in [4] to integrate knowledge delivered from agents.

System determines consensus by choosing optimal classification for each object of environment and placing it in output partition. Optimal classification is the one that generates minimal summary distance to each classification of the same object in profile. Such partition generates minimal summary distance to each of input partitions. Thus it is desired partition and can be presented as an output.

2.2. HIGH INCONSITENCY OF KNOWLEDGE

If processed knowledge is highly inconsistent the system returns unsatisfactory result. In such case there is a need of improving output quality. Humans tend to separate inconsistent opinions into a few cases. Therefore our system acts alike. Clustering techniques are used to split input partitions into several clusters. For each cluster separate consensus is found. All of them are presented as a result.

3. METHOD FOR FINDING CONSENSUS

Finding consensus is an essential task in knowledge integration process. Consensus is an average from statements that come from agents. It is the best opinion satisfying selected evaluation function. Postulates and theorems for consensus functions have been presented in [4]. Form of consensus depends on knowledge structure. The specific way of finding consensus depends mostly on expectations towards output result. Several systems require various conditions to be fulfilled by the consensus.

3.1. KNOWLEDGE STRUCTURE

Input data have structure of semi-hierarchical partitions. They can be incomplete or inaccurate. Observer agents gather data about environment objects. As they classify separate object, they categorize it by examining its attributes. This process is ceased when either all attributes are successfully determined or agent is unable to examine any. Therefore value of all attributes that were more detailed than last determined remains unknown. Agents can make mistakes, so presented classification may happen not to be accurate. Such classified object is placed in a tree. Opinion received from an observer agent contains all classifications the agent was able to determine. Detailed structure is defined in [2]. It is a tree graph where each node represents a class described by set of attributes. Root is the most general classification. Each leaf is the most detailed. Depending on accuracy of classification a single object can be placed either in a leaf of tree or in one of higher nodes. Thus opinion created from such classifications has a form of semi-hierarchical partition. Such opinion is presented as tree *t* and defined in a following way:

Let us assume that:

$Ag = \{ag_1, ag_2, \ldots, ag_Q\}$ is a set of observer agents,

$O = \{o_1, o_2, ..., o_N\}$ is a set of environment objects,

$A = \{A_1, A_2, ..., A_M\}$ is a set of domains of attributes,

$A_m = \{A_m^1, A_m^2, ..., A_m^I\}$ is a domain of attribute m,

Classification of object o_n received from agent ag_q is:

$A(ag_q, o_n) = (a_1^{n,q}, a_2^{n,q}, ..., a_M^{n,q})$, where $a_m^{n,q} \in A_m \cup \{null\}$,

It is assumed that

$a_m^{n,q} = null \rightarrow a_{m+1}^{n,q} = null$

then single input data (an agent's opinion) is

$t = \{A(ag_q, o_n): n = (1, 2, ..., N)\}$

and thus set of opinions from all agents creates profile X.

$X = \{t_1, t_2, ..., t_Q\}$.

Depending on profile consistency, the consensus (output) is either one tree with structure exactly the same as input opinions or a set of such structures.

3.2. METHOD FOR DATA INTEGRATION

In order to integrate knowledge profile into consensus the distance between two opinions needs to be determined. In our system the distance between two opinions is defined as a sum of distances between corresponding object classifications in these two structures. The distance between two objects has been described in [1] and is defined as sum of link weights between one object and another. The weight of a link is $1/d$, where d is natural number describing the depth of a link. The one closest to root has weight of 1. It can be described by following formula:

Let $o_n \in O$ be environment object that has been classified by two agents ag_i, ag_j.

Let $k_i = \max\{k : a_k^{n,i} \neq null\}$, $k_j = \max\{k : a_k^{n,j} \neq null\}$

and $z = \min\{k : a_k^{n,i} \neq a_k^{n,j}\}$

then:

$$d(A(ag_i, o_n), A(ag_j, o_n)) = \sum\nolimits_{v=z}^{k_i - 1} \frac{1}{v} + \sum\nolimits_{v=z}^{k_j - 1} \frac{1}{v} \tag{1}$$

is a distance between classifications of object o_n received from agent ag_1 and ag_2.

When one opinion doesn't contain classification of object for which the distance is calculated, the distance value is 0. Such solution doesn't cause lack of information to decrease consensus precision.

In our system the consensus c is an opinion in which all objects are categorized that generates minimal summary distance to all opinions in profile.

Let $d(t, X)$ be the summary distance between tree t (as defined in 3.1), and all other trees in profile X (also as defined in 3.1).

$$d(t, X) = \sum_{q=1}^{Q} d(t, t_q)$$

$$c = \arg\min_{t \in U} (d(t, X) \qquad (3)$$

where U is an universe of all possible opinions about current state of the environment.

Minimizing squares of sum of distance is thought to be more accurate, but increases the complexity dramatically. This is why we have chosen to use the linear approach. Only such solution allows creating consensus by putting into structure those classifications which minimize the summary distance to their corresponding objects in profile. It is certain that structure created only from objects generating minimal summary distance to their equivalents also generates minimal summary distance to all opinions in profile. We take advantage of linearity of our functions which makes system work fast even when coping with large number of objects.

4. METHOD FOR COPING WITH INCONSITENCY

It's not unusual that central agent receives inconsistent knowledge from observer agents. In such a situation she divides input partitions into several clusters and finds consensus for each of them separately. After this procedure more detailed outcome can be generated, because data does not contradict each other anymore.

The method for choosing optimal number of clusters is presented. If the cluster count is too low, central agent will be unable to create a valuable result because of the inconsistency. On the other hand, if the agent splits data into too many clusters, knowledge would be too shuttered too present any significance.

4.1. DATA CLUSTERING

In our system K-means algorithm is used for clustering input data. This method has been chosen for this task because K-means has proven itself to be worthy and accurate clustering algorithm in many fields of science. It's also fast even for a great number of input partitions. Method is precisely described in [5].

K-means works in our system as follows: as central agent is clustering data into *k* clusters, in first step she selects k random input partitions. Clusters are made from each partition that has the smallest distance to specific center from every other. In next step centroids of each cluster becomes its center and new clusters are created. Those steps are repeated until shifts of centers are below previously set threshold.

Distance between any two partitions is a sum of distances between every two corresponding objects, as defined in (1).

4.2. FINDING THE RIGHT AMOUNT OF CLUSTERS

In order to decide whether clustering process is necessary or not the consistency of profile is calculated. Consistency is rated as follows:

$$p(c, X) = \frac{card(X)}{card(X) + d(c, X)}, \tag{4}$$

is consistency measure of profile *X*, with consensus *c*. The clustering initiates, if consistency is lower than *v* (determined for each particular case).

$$p(c, X) < v. \tag{5}$$

The measure has been introduced by us, and is thought to be satisfactory because it returns value from (0; 1] and makes it easy to determine desired threshold of consistency. However this consistency cannot be used for more than one cluster in profile. It can only be used to determine whether the clustering is necessary or not.

In our system a Silhouette method for validation quality of clusters is used. Method is presented in [7]. In our case central agent performs data clustering starting from *k* = 2 and increasing this value by one as long as silhouette index rises. The right amount of clusters is one before the silhouette index has dropped for the first time.

Final result can be presented after determining optimal clusters number. For each of them consensus is found and thus all of generated partitions are outcome we looked for.

5. FRAMEWORK PROTOTYPE

Prototype framework that realizes task defined in paragraph 3 and 4 has been designed and developed. Artificial environment has been created. Domain has been chosen for several attributes, among which order is important. Random values from domain have been generated to describe objects in environment. Agents have been created with ability to do their task. Interface is provided to control agents and objects gents within environment.

5.1. AGENTS WITHIN FRAMEWORK

In our prototype system two types of agents can be distinguished. Observer agents are designed to gather knowledge about environment. Objects are categorized by them and data is stored in semi-hierarchical tree. Agents don't see all objects and thus don't have complete knowledge on environment. They also aren't unerring, so classifications they generated can be incorrect or incomplete.

The assumption was made that observer agents that are split into 2 groups. First one consist of agents that are "accurate" and the second one can be imagined as agents that were "calibrated wrongly" and thus their opinions are focused on different attributes values than opinions of accurate agents. Within each group knowledge is fairly consistent. In example it has been assumed that there are 10 observer agents and 6 of them were "accurate".

The second type of agent is central agent. There is only single agent of that type that works in the system. She is responsible for knowledge integration. As system notifies request for current state of environment she asks observers for data, conducts integration process and then provides result.

5.2. GENERATED INPUT DATA

Values of attributes of objects in environment have uniform distribution. In example 4 domains were created, each containing 5 possible values. Once values of attributes are set, they can't be changed.

It has been assumed that while conducting observations agent has 70% that she can give any opinion about an object (it can be imagined that she doesn't see 30% of object at all).

Agents then try to determine value of attribute one by one, but they have 4, 8, 12 and 16% that they won't observe attribute respectively on 1st, 2nd, 3rd and 4th level. Generally speaking, the greater chance that they won't see attribute the more detailed it is. Once she doesn't see an attribute procedure stops and classification of object in incomplete.

If the attribute is successfully noticed, observations has binomial distribution with parameters $p = 0.95$ and mean of the correct value.

Those are assumptions that make knowledge inconsistent. As central agent gets partitions made from observations that were conduct in described way, she needs to integrate them.

5.3. DATA INTEGRATION WORKFLOW

On the beginning environment is initialized as described in introduction to this paragraph. Objects are created and random values of their attributers are set. Central agent and observer agents are created. Observer agents start to gather knowledge about environment. As soon as this process is completed central agent asks observers for data. Central agent then determines consensus of gathered profile and measures its consistency using (4). If the result is satisfactory, outcome is provided.

In case consistency is unsatisfactory, clustering process based on *K*-means algorithm begins as described in paragraph 4.2 and right number of clusters is found using Silhouette method, as described in paragraph 4.3. Consensus is determined for each cluster and all of them are provided as result.

6. EXEMPLARY RESULTS

Let us assume that environment consist of 10 objects: o_1 to o_2. There are 4 attributes that describe them and thus 4 different domains were created (As defined in paragraph 3.1), each containing 5 possible values: $(\{A_0^0, ..., A_0^4\}, ..., \{A_3^0, ..., A_3^4\})$. In an exemplary run values were generated as follows:

$o_0\,(A_0^0, A_1^3, A_2^0, A_3^2)$
$o_1\,(A_0^2, A_1^2, A_2^0, A_3^1)$
$o_2\,(A_0^3, A_1^2, A_2^2, A_3^4)$
$o_3\,(A_0^3, A_1^3, A_2^0, A_3^1)$
$o_4\,(A_0^2, A_1^3, A_2^1, A_3^1)$
$o_5\,(A_0^0, A_1^3, A_2^4, A_3^0)$
$o_6\,(A_0^0, A_1^2, A_2^2, A_3^1)$
$o_7\,(A_0^1, A_1^4, A_2^0, A_3^4)$
$o_8\,(A_0^4, A_1^3, A_2^4, A_3^1)$
$o_9\,(A_0^1, A_1^0, A_2^2, A_3^1)$

During classification agents made some mistakes and their classifications were incomplete. For 6 agents in „accurate” group and 4 in „calibrated wrongly” following results were obtained:

s_1 = ---
$s_2 \approx 0{,}8$
$s_3 \approx 0{,}7$
$s_4 \approx 0{,}6$
$s_5 \approx 0{,}5$
$s_6 \approx 0{,}4$
$s_7 \approx 0{,}6$
$s_8 \approx 0{,}7$
$s_9 \approx 0{,}8$

It can be noticed that Silhouette index drops after two clusters so final outcome consists of two classifications. Each one of them is a consensus of different cluster:

$o_0\ (A_0^0,\ A_1^3,\ A_2^0,\ null)$
$o_1\ (A_0^2,\ A_1^2,\ A_2^0,\ A_3^1)$
$o_2\ (A_0^3,\ A_1^2,\ null,\ null)$
$o_3\ (A_0^3,\ A_1^3,\ A_2^0,\ null)$
$o_4\ (A_0^2,\ A_1^3,\ A_2^1,\ A_3^1)$
$o_5\ (A_0^0,\ A_1^3,\ A_2^4,\ A_3^0)$
$o_6\ (A_0^0,\ A_1^2,\ null,\ null)$
$o_7\ (A_0^1,\ A_1^4,\ A_2^0,\ A_3^4)$
$o_8\ (A_0^4,\ A_1^3,\ A_2^4,\ A_3^1)$
$o_9\ (A_0^1,\ A_1^0,\ A_2^2,\ A_3^1)$

$o_0\ (A_0^2,\ A_1^0,\ A_2^2,\ A_3^4)$
$o_1\ (A_0^4,\ A_1^4,\ A_2^2,\ A_3^3)$
$o_2\ (A_0^0,\ A_1^4,\ null,\ null)$
$o_3\ (A_0^0,\ A_1^4,\ A_2^2,\ A_3^3)$
$o_4\ (A_0^3,\ null,\ null,\ null)$
$o_5\ (A_0^2,\ A_1^0,\ A_2^1,\ null)$
$o_6\ (A_0^2,\ A_1^4,\ null,\ null)$
$o_7\ (A_0^3,\ A_1^1,\ A_2^2,\ null)$
$o_8\ (A_0^1,\ A_1^0,\ A_2^1,\ A_3^3)$
$o_9\ (A_0^3,\ A_1^2,\ A_2^4,\ A_3^3)$

The similarity of first classification to objects in environment can be noticed, so it is the result from cluster that contained semi-hierarchical trees from “accurate” agents.

7. CONCLUSIONS

A project of a multiagent system for consensus-based knowledge integration has been presented. Integrated knowledge has form of semi-hierarchical partitions. System is effective thanks to use of linear evaluation function. It also deals with profile inconsistency using K-means algorithm, Silhouette Validation Method, and introduced consistency measuring function. Thus, it can work out highly useful consensus in relatively short period of time, despite large number of environment objects or high data inconsistency.

ACKNOWLEDGEMENTS

This paper was supported by Grant no. N N519 444939 funded by Polish Ministry of Science and Higher Education (2010–2013).

REFERENCES

[1] DANILOWICZ C., NGUYEN N.T., JANKOWSKI L., *Methods for Choice of Representation of Agent Knowledge States in Multi-Agent Systems*, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2002.

[2] KATARZYNIAK R., SKORUPA G., ADAMSKI M., BURDKA L., *A Multiagent System for Consensus-Based Integration of Semi-Hierarchical Partitions. Theoretical Foundations for the Integration Phase*, Semantic Methods for Knowledge Management and Communication, Studies in Computational Intelligence, Vol. 381, 2011, 3–12.

[3] LORKIEWICZ W., POPEK G., KATARZYNIAK R., *Intuitive approach to knowledge integration*. 6th International Conference on Human System Interaction, HSI 2013, Sopot, Poland, June 6–8, cop. 2013, Proceedings [Piscataway, NJ: IEEE], 2013, 1–8.

[4] NGUYEN N.T., *Advanced Methods for Inconsistent Knowledge Management*, Springer-Verlag, London 2007.

[5] HARTIGAN J.A., WONG M.A., *A K-Means Clustering Algorithm*, Applied Statistics, Vol. 28(1), 1979, 100–108.

[6] POPEK G., KATARZYNIAK R., *Interval-Based Aggregation of Fuzzy-Linguistic Statements*. Proceedings of FSKD 2013, 23–25 July 2013, Shenyang, China, IEEE, 2013.

[7] ROUSSEUW P., *Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis*, Computational and Applied Mathematics, Vol. 20, 1987, 53–65.

PART 3

BIG DATA
AND HIGH PERFORMANCE COMPUTING

databases, GPGPU, PostgreSQL, OpenCL

Kamil KSIĄŻEK
Piotr SAPIECHA*

USING GRAPHICS PROCESSING UNIT TO ACCELERATE DATABASE QUERY EXECUTION

One of the major problems in database management systems is handling large amounts of data while providing short response time. Problem is not only proper manner of storing records but also efficient way of processing them. In the meantime GPUs developed computational power many times greater than that offered by comparable CPUs. In our research we investigated benefits that using GPU in database queries execution can give. Using offered in PostgreSQL database User-Defined Aggregate extension mechanism, we implemented our versions of 3 standard aggregates: sum, average and population standard deviation that were executed on GPU by using OpenCL. We found that, while in simple aggregates (sum and average) there was no gain, in more complex aggregates (population standard deviation) we were able to achieve more than 2 times shorter execution time than in standard build in database aggregate. We conclude that processing database queries on GPU can be very effective, provided that proper method of dispatching operations between CPU and GPU is used.

1. INTRODUCTION

The Graphics Processing Unit (GPU) is a processor, which was invented for creating realistic graphics images on computer screen. To meet such a goal GPUs have to have high floating-point operations performance. As a result GPUs quickly outperformed central processing units (CPUs) in fields that benefits its different architecture [1], [2]. This advantage became useful not only in graphics processing but also in performing other computations. It gave a start to a new field of GPUs use – General-Purpose computing on Graphics Processor Units (GPGPU).

In first attempts to use GPUs computing power for general-purpose tasks, data that was meant to be processed, had to be mapped to pixels. GPUs lack of ability to process basic data types like integers and chars was not allowing for GPGPU in a straight-

* Institute of Telecommunications, The Faculty of Electronics and Information Technology, Warsaw University of Technology, ul. Nowowiejska 15/19, 00-665 Warszawa.

forward way. In 2007 there was a major breakthrough – Nvidia made CUDA (Compute Unified Device Architecture) software development kit (SDK) public [1]. From now on GPGPU programmers had a tool that allowed them to program GPUs almost like ordinary CPUs. But there were also limitations. The biggest one was limited GPUs support – CUDA was, and still is, available only for Nvidia GPUs. In the meantime, the biggest competitor of Nvidia on GPUs market – ATI (now part of AMD) released their own GPGPU technology – ATI Stream. It was not very successful, partly because of the next big thing that happened in field of GPGPU. This thing was OpenCL – open standard for general-purpose computation that was designed not only for GPUs [3], [4]. Initially developed by Apple with AMD, IBM, Intel and Nvidia collaboration, this standard freed programmers from necessity to design their programs for specific devices. OpenCL in contrast to Nvidia CUDA and ATI Stream not only allowed to run programs on GPUs from different vendors, but also gave support for other devices like CPUs, Advanced RISC Machine (ARM) processors or recently even on Field Programmable Gate Arrays (FPGA) circuits.

Another, important for this paper, process is development of Databases Management Systems (DBMSs) servers architecture. Recently we can observe that data is less and less often stored locally. It is caused by development of cloud computing and by security and reliability reasons. On the other hand, it gives database architects great opportunity to improve DBMSs performance [5], [6]. Having multiple sources of data allows to decrease negative influence of slow I/O operations on widely used Hard Disk Drives (HDDs). Furthermore it becomes more cost effective to store data on faster data storages. Solid-State Drives (SSDs) for example, offers more data throughput than HDDs and their prices becomes more and more acceptable [7]. Also Random Access Memory (RAM) sizes are growing more rapidly than sizes of databases, what encourages to store on them not only data processed at the moment. In such case bottleneck in I/O operations on data source is practically completely eliminated. As a result DBMS performance is limited only by client-server communication time, RAM data transfer and CPU performance.

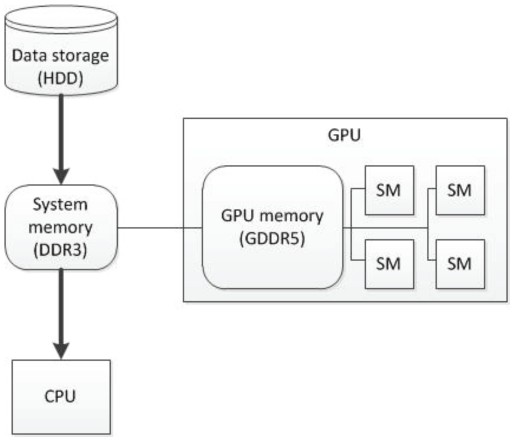


Fig. 1. Data flow in system with CPU processing

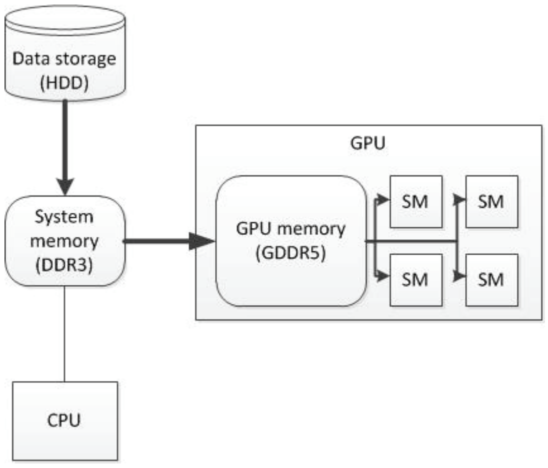


Fig. 2. Data flow in system with GPU processing

2. PROBLEM SPECIFICATION

In this paper we try to contribute to ongoing researches in field of GPU use in DBMSs queries execution. Unlike majority of related researches [8], [9], we examine whole system with data stored on HDD. Furthermore we aimed to perform tests on comparable CPU and GPU using DBMS with very little modifications in its structure. This is supposed to give fair comparison and examine possibilities of GPU use in DBMSs without major modifications in them that could cause compatibility issues. Taking into account that we used PostgreSQL database in our research, this paper focus particularly on possibilities that this specific DBMS gives for GPU computing.

Because of large range of queries types that current DBMSs can perform, we decided to narrow down our research to 3 specific queries counting respectively: sum, average and population standard deviation of values given in pointed column. Whole operation takes certain column from pointed table as an input and returns single value, which is result of previously enumerated tasks. Although efficient methods to perform these queries on CPU are well known and already implemented, we changed data flow inside them. As a result we can exploit parallel character of these queries and thanks to GPUs properties achieve better computing time results.

Similar researches were already done [8]–[10] but usually they focus on in-memory DBMSs and assume that database is stored in system memory for CPU or in GPU memory for GPU processing [8], [9]. In such cases influence of low transfer speed to and from HDD is completely ignored. Also in these researches [8], [9] is not discussed fact that single CPU can has much more system memory than comparable GPU, which is very important if we would like to store whole database in memory (currently GPUs has no more than 4 GB of GDDR5 memory). Off course we can store data for GPU processing in system memory, but in this case we would have to read them through PCI Express, which is much slower than reading from GPU memory and would have decreased achieved results. In contrast to these approaches [8], [9] we decided to examine GPU behavior in whole system, which shows not only HDD but also PCI Express influence and allows using our programs in DBMSs, which sizes are not limited by system/GPU memory.

3. OUR APPROACH

The final shape of our solution was dictated by two assumptions. First of all we wanted to perform our computations on widely available (and what comes with that, not expensive) GPU. For the fairness of the comparison CPU was from the same price range. Second assumption was an idea to add GPU support as an extension to existing DBMS. We didn't want to change whole DBMS because it would result in limiting its current capabilities.

3.1. HARDWARE SPECIFICATION

In presented tests we used ordinary PC with following configuration:

- CPU – AMD Athlon X3 450
  - 3.2 GHz clock frequency
  - 384 KB L1 memory
  - 1.5 MB L2 memory
  - Up to 37.3 GB/s memory bandwidth
- GPU – Nvidia GeForce GTS 450
  - 4 Streaming Multiprocessors (SM) with 48 CUDA cores each
  - 1566 MHz clock frequency
  - 1 GB GDDR5 memory
  - Up to 57.7 GB/s memory bandwidth
  - 16 GB/s PCI-E 2.0×16 bandwidth
- System memory
  - 2×2 GB memory
  - DDR3 1333 MHz
- HDD
  - WDC WD2500KS-00MJB0
  - 250 GB capacity
  - 53 MB/s average transfer rate

3.2. OPENCL STANDARD

All presented test cases benefit from parallel computing thanks to OpenCL standard [3], [4]. The choice of OpenCL over CUDA is justified by larger range of compatible devices. Thanks to this fact we can achieve much more hardware independent implementation that with little to no modifications can be used on many other devices than GPU used in tests. As previously mentioned, OpenCL is open standard – its implementation is hardware vendor dependent. In this research OpenCL implementation provided by Nvidia was used [11].

Main advantage that OpenCL gives is possibility to use data and instruction parallelism that specific device has to offer. In GPU case it allows to manage hundreds of available CUDA cores, achieve higher memory bandwidth with coalesced memory operations, getting access to all available memory resources (including on-chip memory) and queue write/read operations between system and GPU memory.

All this functions can be achieved thanks to OpenCL programing model. One of its parts is execution model, which consists of work-items (threads executed on single core) and work-groups (groups of threads executed on single processing unit). Another part is memory model, which defines global, constant, local and private memory spaces that differs in performance and access range.

3.3. POSTGRESQL DATABASE

We chose to perform our tests on PostgreSQL DBMS mainly because of its open character and good range of extensions mechanisms. In this paper we focused on one of those mechanisms – User-Defined Aggregate (UDA). It allows us to implement our own aggregate function, which is a function that group together values of multiple rows and returns single value.

Definition of UDA consists of 3 main elements:

1. Type that will contain aggregate states between function calls
2. Function that will be called for each row of input data. It can perform computations and store their results in previously mentioned state type.
3. Final function that will be called after every row of input data is processed. This function is optional – some aggregates like sum can be created without this function.

3.4. SOLUTION IMPLEMENTATION

In our work we implemented our own UDA, which functions were written in C with OpenCL libraries. As a result, we got almost fully operative aggregates that could be called from ordinary SQL queries. Whole process of defining our UDA can be divided in following steps:

1. Defining of previously mentioned UDA elements: state type ("stype"), function called for each row ("sfunc") and final function ("finalfunc"). Whole code was created according to PostgreSQL documentation.
2. Adding OpenCL support by adding proper libraries.
3. Adding proper fields in "stype" for storing row values.
4. Modifying "sfunc" to store row values in "stype".
5. Creating whole OpenCL program in "finalfunc" that perform aggregate function on data previously stored by "sfunc" in "stype".

Using this method we implemented 3 aggregates that can be found in PostgreSQL 9.1 DBMS – sum, average (avg) and population standard deviation (stddev_pop).

As a result we created aggregates that in their sfunc store read input values in stype. No other operations are performed on this stage, which can be considered a drawback in case of simple aggregates, that can be computed between data reads. Such approach dictates structure of stype. It consists of array that holds values that were read in sfunc and single integer that holds number of records written to mentioned array. When all input data are stored in stype, finalfunc is called. Here are performed all procedures required to run OpenCL kernel: getting platform and device information and creating context, command queue, buffors, program and kernel. Also input data are copied from stype to GPU memory. After that OpenCL kernel that executes aggregate operation is performed.

Kernel uses separate OpenCL work-item for each input table row. During our research we tested few different approaches to kernel implementation. We examined influence of using multiple work-items, multiple work-groups and local memory. For these initial tests we created multiple implementations of sum aggregate:

1. single-threaded, global memory using sum,
2. single-threaded, local memory using sum,
3. multi-threaded in a single work-group, global memory using sum,
4. multi-threaded in 4 work-groups, global memory using sum,
5. multi-threaded in a single work-group, local memory using sum,
6. multi-threaded in 4 work-groups, local memory using sum.

As a result we determined that fourth implementation seems to be most suitable for our purposes. Work-items are stored in 4 work-groups – containing 250 work-items each. In our research we discovered that in these particular operations using local memory in kernel increases its execution time. Much better option is to leave local memory to be used by GPU as a cache memory, which was implemented in Fermi architecture [2].

Sum aggregate kernel starts with initializing result field in GPU global memory with 0 value. After that synchronization barrier is used to make sure that this operation is completed before starting next one. When result field is ready each work-item performs atomic add operation on single input data value and result field.

$$\sum_{i=1}^{N} x_i$$

Average aggregate kernel is very similar to the previous one. Only difference is that after summing all input data result is divided by number of work-items, which corresponds to number of summed values. To make sure that sum operation is completed before divide another barrier is used.

$$\mu = \frac{\sum_{i=1}^{N} x_i}{N}$$

Last aggregate, population standard deviation, is the most complex one. After initializing variables in a similar way that it was done in previous kernels, each work item performs two sums: one for given input data value and the other for its square. In this place, similar to average aggregate kernel, we use barrier to make sure that operations are completed. After that we calculate population standard deviation by putting received sums to the given formula:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N} x_i^2}{N} - \left(\frac{\sum_{i=1}^{N} x_i}{N} \right)^2}$$

4. COMPUTATIONAL RESULTS

Because of many factors that influence query execution time, to get fair results we executed ours and standard aggregates 10 times each. Input data was a table with 1000 rows filled with integers.

Table 1. PostgreSQL 9.1 standard aggregates execution times

Query	**sum()**	**avg()**	**stddev_pop()**
Average time [ms]	1,4748	1,634	7,0842

During tests of our own UDA we observed much longer execution times than in standard aggregates. After examining which part of code cause this slow down, we found out that OpenCL context creation can last even up to 50 ms. Because context can be initialized earlier and stored in database in Table 2. are presented results reduced by context creation time. Also Table 3. shows times of processing given query strictly on GPU.

Table 2. PostgreSQL 9.1 UDA with OpenCL support execution times reduced by context create times

Query	**sum() time**	**avg() time**	**stddev_pop() time**
Average time [ms]	2,4744	2,4882	2,8239

Table 3. PostgreSQL 9.1 UDA with OpenCL support GPU processing times

Query	**sum() time**	**avg() time**	**stddev_pop() time**
Average time [us]	23,9264	24,896	36,272

5. DISCUSSION

To conclude results we present gain in execution time that can be achieved by using OpenCL in UDA (Table 4). It was calculated by dividing difference between standard aggregate execution time and UDA execution time by UDA execution time. While sum and average UDA were slower, the last aggregate is over two times faster than standard one. Worse results of first two queries can be explained with negative influence of HDD data transfer speed. Both sum and average does not require many computations to be performed. Also these computations can be done between reading succeeding rows, which give CPU major advantage. In the last aggregate, which is also most complex one, we can observe big gain in execution time, even though HDD input/output operations still have major influence on the results.

Table 4. Gain in time execution achieved by using UDA instead of standard aggregates

Query	sum()	avg()	stddev_pop()
Gain	–40%	–34%	151%

Impact that reading and writing data from HDD has on DBMS performance, can be seen when we compare whole queries execution times (Table 2) with time it takes to perform aggregates operations inside GPU (Table 3) – please notice that times in Table 2 are in milliseconds while in Table 3 we have results in microseconds. That leads to a conclusion that processing time alone is only about 13% of whole query execution time in case of using GPU. This shows that true benefits of using GPU in DBMS queries execution can only be achieved with respectively fast data source.

As previously mentioned, our work has major differences from majority of researches [8], [9] in field of using GPUs in DBMSs. Taking under account fact that in these works data source is approximately even 2000 times faster (for [8] where GPU memory was used, max transfer is over 100 GB/s, while we used HDD with average transfer of 53 MB/s) comparing our results with them would be unfair. On the other hand similar conditions for good comparison can be found in [10]. In this work also PostgreSQL DBMS was used and data were stored in HDD (but no information about its transfer speed was given). On the contrary performed query was different (counting spatial distance between points) and it was realized using different PostgreSQL mechanism, so comparison results should not be treated as decisive. In this research, in the best scenario 400% gain was achieved, but only for input data range of approximately 100000–1000000 rows. For our scenario, where we used 1000 rows, in discussed paper GPU algorithm was about 2 times slower than build-in function, when we achieved aggregate function over 2 times faster from its built-in equivalent. It suggests that our solution can achieve even better results for larger input data sets, which should be done in future research.

6. CONCLUSION

Accelerating DBMSs with GPUs is a very difficult challenge, which we wanted to achieve without completely changing DBMS structure. Unlike other works [8], [9] we decided to not give the GPU advantage by using its memory as data source and changing data structure so it would be easier for GPU to process. Thanks to that, our solution let us achieve major performance improvement in some cases (up to over two times shorter execution time than on CPU) while still keeping all of DBMS functionalities.

REFERENCES

[1] *CUDA C Programming Guide Version 4.2*, NVIDIA Corporation, 2012.
[2] *NVIDIA's Next Generation CUDA Compute Architecture: Fermi*, NVIDIA Corporation, 2009.
[3] SCARPINO M., *OpenCL in Action: How to accelerate graphics and computation*, Manning Publications Co., 2012.
[4] *The OpenCL Specification*, Khronos OpenCL Working Group, Version: 1.1, Document Revision: 44, 2011.
[5] MUELLER R., TEUBNER J., ALONSO G., *Glacier: A Query-to-Hardware Compiler*, Proc. of the ACM SIGMOD Conference on Management of Data (SIGMOD 2010), Demonstration, Indianapolis, IN, USA, June 2010.
[6] *ParStream – Turning Data Into Knowledge*, White Paper, ParStream, 2010.

[7] DO J., DEWITT D., ZHANG D., NAUGHTON J., PATEL J., HALVERSON A., *Turbocharging DBMS buffer pool using SSDs*, Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, Athens, Greece, June 2011.

[8] BAKKUM, P., SKADRON, K., *Accelerating SQL database operations on a GPU with CUDA*, Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units, GPGPU 2010, ACM, New York, 2010, 94–103.

[9] HE B., LU M., YANG K., FANG R., GOVINDARAJU N., LUO Q., SANDER P., *Relational Query Co-Processing on Graphics Processors*, TODS, December 2009.

[10] APTEKORZ M., SZOSTEK K., MŁYNARCZUK M., *Możliwości akceleracji przestrzennych baz danych na podstawie procesorów kart graficznych oraz funkcji użytkownika*, Studia Informatica, Vol. 33, 2012, nr 2B, 145–152.

[11] *OpenCL Programming Guide for the CUDA Architecture Version 4.2,* NVIDIA Corporation, 2012.

*natural language processing, error detection,
automatic text corrections, dependency grammar*

Dariusz BANASIAK*, Jarosław MIERZWA*,
Antoni STERNA*

DETECTION AND CORRECTION OF ERRORS IN POLISH BASED ON DEPENDENCY GRAMMAR

The paper presents an approach to computerized detection and correction of errors in Polish texts. Most contemporary editors provide solutions that prove ineffective if misspelled word transforms into another, valid word. Resulting sentence may be grammatically or semantically incorrect, even though all words are correct. This phenomenon is especially common in Polish due to rich inflection, complex case system and the use of diacritical marks. Error detection and disambiguation among correction candidates may require context-dependent analysis, based on syntax or even semantics. To introduce effective methods of error detection, typical causes of errors and possible error patterns are considered. Proposed method of error detection and correction is based on suitably adapted Dependency Grammar. It allows to capture syntactic and, partially, semantic dependencies in sentence, even between distant words or phrases. Therefore, inflectional properties of linked words may be analyzed and best word selected form the list of correction candidates. Presented approach may be used as supplement tool in automated text processing systems.

1. INTRODUCTION

Nowadays computers are used extensively in text documents preparation. Errors in text appear mostly when digital form of text emerges as a result of typing, optical character recognition (OCR) or voice recognition (speech-to-text). Further operations on digitalized text (e.g. reading, writing or transmission) have relatively good protection mechanisms and usually do not introduce additional errors. Therefore, it is essential to detect and correct errors as early as possible to prevent them from spreading in web-based information systems.

Although all commercial text editors include good spelling checkers, they are limited in their scope and accuracy. Correction techniques focus mainly on isolated word, ig-

* Institute of Computer Engineering, Control and Robotics, Faculty of Electronics, Wrocław University of Technology, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland.

noring linguistic context in which that word appears. Furthermore, the list of proposed corrections for non-word (i.e. word not found in dictionary) is sometimes extensive and selection is left to the user. This approach is not acceptable in applications, such as speech-to text or OCR, which are required to perform unsupervised error detection and correction. To the best of our knowledge, there are no comprehensive solutions for general domain texts that contain multiple errors of both syntactic and semantic nature.

The problem of error detection and correction is especially complicated in Polish, due to the following features of language:

– the presence of diacritical marks, which results in frequent letter substitution (typically by omission of diacritical marks),
– rich inflection that results in many word forms,
– complex case system imposing many requirements on the form of related words.

2. TYPES OF ERRORS AND THEIR SOURCES

Text correction is a process of linguistic analysis which should trace and eliminate possible wording errors. This task, traditionally performed by human proof-reader, should be automated, with minimum intervention from the user. To develop effective methods of error detection, the nature of errors occurring in texts should be considered. Typical errors found in Polish texts are presented below:

– spelling and lexical errors (wrong word usage),
– typographical errors (wrong letter, usually keyboard related),
– syntactic errors (improper collocations of word forms),
– inflectional errors (inappropriate word form),
– stylistic errors (improper selection of language resources in given statement),
– punctuation errors.

Typical causes of errors include insufficient language knowledge and careless or hasty typing. In present work errors introduced during text edition are considered. In Polish, due to the method in which diacritical marks are obtained (Alt + letter), diacritics related errors are usually predominant. Tests show, that omission of diacritical mark is common whereas redundant marks are relatively infrequent [1].

Some errors appearing during text edition (spelling and typographical errors) may be eliminated by automatic correction facilities, available in most present-day text editors. However, it should be noticed that methods currently used in text editors (especially for Polish) are not very sophisticated. They consist mainly in verification if words in texts are present in dictionary or can be found in list of errors. Such simple methods prove ineffective if after edition all words in sentence are correct and yet entire sentence is incorrect (e.g. due to syntactic or inflectional errors). Let us consider example sentence (1).

Poszedłem z kolega do domy.

(I went home with my friend.) (1)

In this example all words forming the sentence (1) are correct since they are present in dictionary. However, this simple sentence contains two errors because the forms of words *kolega* and *domy* are inappropriate. The first error is the result of Alt key omission at last letter. The second error is caused by substitution of "*u*" for "*y*", the character adjacent on the keyboard (the correct form of word in this context is *domu*). To detect (and subsequently correct) such kind of errors it is necessary to apply advanced methods of natural language processing.

3. ERROR DETECTION

Example sentence (1) demonstrates that error detection is essential, preliminary step in automatic text correction. Analysis of real-world texts (newspaper articles, websites, master's thesis and so on) provides an abundance of similar examples. It should be noted that even simple error, made during text edition (e.g. letter omission, extra letter, letter transposition, omission of Alt key, resulting in lack of diacritical mark) may lead to significant change of the form or even meaning of word. The fact is of great importance for error detection. The following cases may be distinguished:

– erroneous sequence of letters does not form any correct word,
– error leads to the change of word form, e.g. *domu – domy*,
– the word form is preserved but its meaning changes, e.g. *listy – lisy*, *klasa – kasa,*
– distorted word belongs to different lexical category than its correct prototype, e.g. *więc – wiec*, *noże – może*, *lub – klub*.

Contemporary text editors cope well with the first type of errors (at least with regard to detection). Therefore, such errors will not be discussed in present work. To detect remaining types of errors advanced linguistic knowledge is necessary, both about grammatical structure of sentence (syntactic analysis) and meaning of individual words and entire sentences (semantic analysis).

Computer linguistics provides many useful methods and tools for automatic processing of natural language texts. Some of these methods may be applied to error detection. They may be divided into the following groups:

– methods based on statistical analysis of words,
– methods based on context of words,
– methods using grammars to check correctness of the sentence*,*
– methods based on meaning analysis of individual words or entire sentence.

Statistical methods play important role in computer linguistics. They make extensive use of language corpora, available in digital form. Significant research area is analysis of vocabulary used in real-word texts. This kind of research resulted in "Frequency Dictionary of Polish" (*Słownik frekwencyjny języka polskiego – SFJP*) [2]. The list of 10 most frequently used Polish words is presented in Table 1.

Table 1. Example fragment of SFJP dictionary

Word	Number of occurrences
się	2,844,816
i	2,658,222
w	2,477,368
nie	2,209,859
na	1,928,331
z	1,725,331
do	1,239,463
to	1,202,756
że	1,129,684
a	789,895

The frequency of given word in documents may be helpful criterion in selection of potentially erroneous words. It may be assumed that word with lower rate of occurrence in Polish texts is more probable error candidate.

Another factor, useful in error detection at the level of individual words, is the susceptibility of given word to transformation into another, correct word. On the assumption that errors in text appear as a result of common, usually typographical mistakes (letter omission, extra letter, transposition of adjoining letters etc.) it is possible to specify the list of correct words that may be obtained as a result of transformation. Longer list indicates higher word susceptibility to errors which are hard to detect. For instance, shorter words usually have more substitutes than longer words. Obviously, words sensitive to deformation are another potential cause of errors.

Among statistical methods N-gram analysis is worth noticing because it encompasses close context of words. Techniques based on N-grams may be used in the stage of error detection as well as correction. N-grams allow predicting the next word in the sequence (or at least its lexical category). The range of analyzed context depends on the length of N-gram. It is directly preceding word for bigrams, two words for trigrams and three preceding words for tetragrams. N-gram techniques examine N-grams in input string and search precompiled table of N-gram statistics to get information on its frequency. To create precompiled table of N-grams large resources of linguistic are necessary. In case of Polish publicly available corpora may be used [3], [4].

An example of practical application, based on of N-grams, is presented in [5]. The Linguistic Habit Graph (LHG), built from trigrams gathered in Polish websites, reflects word dependencies in real-word texts. The system checks entered words, suggests correct word or its ending and performs automatic, contextual correction.

4. LINK GRAMMAR

Link Grammar [6] belongs to the class of Dependency Grammars which represent relationship between words in sentence. Dependency structures abstract away from linear order, therefore they are well suited for Polish, the language with free word order. In Link Grammar relation between two words is represented by labelled link, the name of label reflects the type of relation. Each link is a combination of two connectors attached to words. Each connector has its owner (the word from which the connector originates), direction (left or right) and the name called label. For convenient processing Link Grammar may be expressed in disjunctive form presented as expression (2).

$$((L_1, L_2, ..., L_m)\ (R_n, R_{n-1}, ..., R_1)) \qquad (2)$$

As is shown in expression (2), each disjunct includes two ordered lists which contain the sequence of connector names (L_i in the left list and R_i in the right list). Each word has a set of associated disjuncts. The sets of disjuncts for individual words are stored in dictionary (denoted by symbol L). This sets describe linking capabilities of the word, i.e. specify how that word may be linked with other words.

To build the sentence structure first the sets of connectors for each word are extracted from dictionary L (the grammar is defined there). Then an attempt is made to build the graph structure (by combining connectors into the links). Connectors originating from two different words may be joined and consequently create the link if they have compatible labels and opposite directions. The process may be referred to as connector matching. According to the rules of Link Grammar resulting graph (including a set of links also called a linkage) represents correct sentence if the following conditions are satisfied:

- the graph in which words play the role of vertices and links act as edges is connected (the set of links connects all the words together),
- the graph is planar (i.e. links drawn above the words do not cross),
- linking requirements for all words are satisfied (i.e. all connectors from selected disjunct take part in the process of linking).

5. LINK GRAMMAR FOR POLISH

Original Link Grammar was designed for syntactic analysis of English. This lexical type grammar (linking requirements are specified for individual words) has all advantages and shortcomings specific to the approach. In order to adapt Link Grammar to Polish, many aspects peculiar to the language should be considered. Accommodation represents typical inflection related phenomenon in Polish [7] (it is not present in positional languages such as English). It strongly affects the process of connector matching; additional requirements must be met when links are created.

Polish words have basic forms and some features that are called inflectional categories. Typical examples of inflectional categories are case, number and gender. Each inflectional category assumes value from specific set (e.g. for case the set includes the following values: nominative, genitive, dative, accusative, instrumental, locative and vocative). Each lexical category has its own set of inflectional categories (e.g. for noun: case, gender, number and person). As was mentioned before, in original Link Grammar linking requirements are specific to individual words. In proposed modification for Polish it is assumed that linking requirements are ascribed to lexical categories and subcategories.

6. CONNECTOR MATCHING

In English version of Link Grammar each connector consists of base name (the sequence of upper case letters) and optional suffix (lower case letter) which represents information about additional requirements imposed on the connector of matching candidate. The concept of compatibility, introduced earlier, may be now formulated more precisely. Connector labels are compatible in following cases:

– both labels are identical,
– base label names are identical and one of connectors has additional suffix.

In Polish extra requirements are imposed on matching connectors. In consequence connectors will be parameterized and connector labels assume the form presented in expression (3).

$$L(k_1, k_2, ..., k_n) \tag{3}$$

In expression (3) L denotes the base name whereas the sequence of symbols k_i represents the list of parameters (inflectional categories).

The process of matching consists of two stages. In first stage connectors are regarded as matched if their labels have the same names and additionally:

– both lists of parameters are identical,
– one list has some parameters and the other is empty.

If the first stage of matching was successful, then following operations are performed in second stage:

– the list of parameters is filled with values, based on word features (each parameter k_i is replaced with specific value),
– additional requirements, imposed on linked words, are checked (e.g. for connector of type P, linking subject with predicate, the value of case should be

nominative). These requirements are stored in dictionary *L* (which also contains the sets of disjuncts as was mentioned before),
- the values of all parameters are compared – they should be compatible for all pairs of inflectional categories.

If all requirements are satisfied and the values of inflectional categories are compatible then the link connecting considered words may be created.

Presented above discussion is merely outline of general idea. More detailed solutions, based on proposed model are presented in [8].

Combining connectors into the links is elementary operation in the process of transformation of sentence into the graph. The process consists of the following steps:
- the sentence is split into words,
- lexical categories and features of words (values of inflectional categories) are extracted from dictionary (e.g. using Morfeusz dictionary [9]),
- the sets of connectors are assigned to words,
- preliminary reduction of the sets of connectors is performed (pruning),
- an attempt is made to build the linkage for each possible combination.

7. CONNECTOR REDUCTION

More than one disjunct may be attached to each word (this stems from the fact that given word may play various roles in sentence, e.g. subject or adverbial). The number of disjuncts for some words may get quite large. To reduce this number the method called pruning was devised in Link Grammar. If given word in the sequence has disjunct containing connector *X* in the right list then some word on the right should have compatible connector in the left list of its disjunct. If such word cannot be found, then considered disjunct is removed as irrelevant in given sequence. Similar operation is performed for the left list of connectors. The series of sequential passes through the words, called pruning steps, is performed alternately, form left-to-right and right-to-left. The process is repeated as long as any reduction is possible.

As was mentioned before, connector matching is performed at two levels. Since the word may belong to more than one lexical category, the sets of connectors are merged before reduction and therefore include connectors derived from alternative categories or subcategories.

The process of reduction may be used to exclude some words. Let us assume that the lists of correction candidates are available for each word. The lists are obtained by word transformations modelling typical errors (e.g. letter omission or letter transposition). If all disjuncts attached to given word are reduced in the process of pruning, this means that the word cannot be used as replacement word in considered context.

8. THE OUTLINE OF DETECTION AND CORRECTION

At the beginning we assume that all words in the sentence have correct forms. To verify this assumption Link Grammar analysis is performed (an attempt is made to build the graph). In case of failure (graph cannot be built) the process of detection and correction is performed in following steps:

– the lists of replacement words (correction candidates) are created (the original word is also included in the list),
– lexical classes (or subclasses) are determined for each word present in the list, joined list of classes are obtained,
– combined set of disjuncts is produced for each word in the sentence,
– the process of reduction is performed (as a result some replacement alternatives may be excluded).

To illustrate the process the analysis of example sentence (4) is presented.

Chłopak poszedł z dziewczyną do domu.
(The boy went home with a girl.) (4)

The sentence (4) is correct and after analysis based on modified Link Grammar the graph presented in Figure 1 is obtained.

```
              +-------------------HP--------------------+
+-P k2,k3,k4--+-----HP------+----WP k1----+             +----WP k1----+
|             |             |             |             |             |
chłopak                     z                           do
              poszedł                     dziewczyną                  domu
noun          verb          preposition   noun          preposition   noun
```

Fig. 1. The result of analysis for sentence (4)

Let as assume now that two errors has been inserted into example sentence (4). The Alt key was not pressed in word *dziewczyną*, as a result the word changed into *dziewczyna*. In word *domy* the key “y” was pressed instead of “u” (such mistake is very likely because of keyboard layout). As a result of these errors the sentence has been distorted into the form (5).

Chłopak poszedł z dziewczyna do domy. (5)

Although all words in the sentence (5) may be found in dictionary, the words *dziewczyna* and *domy* are incorrect in this context. The graph cannot be created for this sentence because connectors will be reduced already at the stage of pruning. Therefore the list of correction candidates, shown in Table 2, will be generated.

Table 2. Correction alternatives for sentence (5)

Chłopak	poszedł	z	dziewczyna dziewczyną	do	domy domu

As a result of pruning the words *dziewczyna* and *domy* will be eliminated from the lists of correction alternatives. The only remaining alternative words (*dziewczyną* and *domu*) allow creating the correct graph shown in Figure 1.

9. APPLICATION OF N-GRAMS

Preliminary reduction of correction alternatives may be also carried out with the help of N-grams. The size of required data structures may be decreased if N-grams are based on lexical categories and subcategories instead of specific words. Statistical information obtained from N-grams may allow reducing:

– the number of categories and subcategories associated with words and in consequence reducing the number of connectors (more than one category may be ascribed to the word although only one is correct in given context). Selection at this stage is used only to eliminate categories with zero probability of occurrence, selection among alternatives with non-zero probability is not performed at this stage,
– the number of alternative words (if all possible lexical categories are excluded).

It is possible that after reduction of categories and connectors still remain some correction alternatives. In such case statistical information, obtained from N-grams, may be used to make decision about the order in which alternatives will be checked during Link Grammar based analysis. The most probable alternatives will be analyzed first.

10. CONCLUSION

Although most text editors are equipped with error detection and correction mechanisms, they focus mainly on isolated words. This significant drawback may result in numerous undetectable errors, especially in languages with complex word dependencies such as Polish. This leads to the conclusion that contextual information should be used to improve detection and correction accuracy.

The paper presents an approach to detection and correction of errors in Polish texts. Proposed method is based on modified Link Grammar, which allows encompassing the aspects specific to Polish, mainly accommodation and inflection. In correction stage the list of possible replacement words is generated. Then the list is reduced to select best candidates, and attempt is made to build the graph representation of the sentence. The details of reduction and selection among correction candidates are the subject of further analysis and experimental work.

REFERENCES

[1] WABIK A., *Detekcja błędów fleksyjnych w komputerowych edytorach tekstu*, M.Sc. Thesis, Wrocław University of Technology, Wrocław, 2010, 5–11.

[2] KGLK Krakowska Grupa Lingwistyki Komputerowej, *Słownik Frekwencyjny Języka Polskiego*, 2009.

[3] PRZEPIÓRKOWSKI A., *The potential of the IPI PAN Corpus*, Poznań Studies in Contemporary Linguistics, Vol. 41, Poznań, 2006, 31–48.

[4] PRZEPIÓRKOWSKI A., BAŃKO M., GÓRSKI R.L., LEWANDOWSKA-TOMASZCZYK B., *Narodowy Korpus Języka Polskiego*, Wydawnictwo Naukowe PWN, Warszawa, 2012, 11–23.

[5] GADAMER M., HORZYK A., *Automatyczna kontekstowa korekta tekstów z wykorzystaniem grafu LHG*, Computer Science AGH, Vol. 10, Kraków, 2010, 39–57.

[6] SLEATOR D.D.K., TEMPERLEY D., *Parsing English with a link grammar*, Technical Report CMU-CS-91-196, School of Computer Science, Carnegie Mellon University, Pittsburgh, 1991, 7–26.

[7] SALONI Z., ŚWIDZIŃSKI M., *Składnia współczesnego języka polskiego*, Wydawnictwo Naukowe PWN, Warszawa, 1998, 108–123.

[8] NIEWIADOMSKI T., *Sprawdzanie poprawności pisowni zdań języka polskiego za pomocą klasycznej gramatyki łączeń*, M.Sc. Thesis, Wrocław University of Technology, Wrocław, 2003, 36–75.

[9] WOLIŃSKI M., *System znaczników morfosyntaktycznych w korpusie IPI PAN*, Polonica XXII-XXII, Kraków, Wydawnictwo LEXIS, 2003, 39–54.

*distributed computing, JavaScript,
browser-based computation*

Jan KWIATKOWSKI*, Maciej DZIK*

PARALLEL DISTRIBUTED COMPUTING USING JAVASCRIPT

Nowadays personal computers became very popular, moreover most of them are connected to the Internet. One can say that all of them together create a powerful distributed environment that can be used for solving problems, which need parallel processing. On the other hand Web technologies are very popular, too. One of its biggest advantages is it's accessibility, it doesn't require any special knowledge to use it and there is a large range of devices that implement it. There are webpages that have millions visitors daily. Therefore the question raises, is it possible to use this technology to get access to the incredible computing power that lays in the devices connected to the computer network that utilize just only a part of their actual power. The chapter describes a framework that utilizes Python and new extensions for JavaScript to delegate code to be executed by a web site clients and gathers results on a server. It means that it provides a fully functional basic environment for employing parallel distributed computing in the WWW environment.

1. INTRODUCTION

A lot of problems that can be found in the real world still cannot be solved in reasonable time, even using the most powerful computers currently available. Especially heavy computations are required in studying biological, chemical and geographical science problems. For example, there are applications that are meant to decipher genome structures, finding cures for currently incurable diseases such as cancer, but also analyze weather and earth characteristics as well as looking for extraterrestrial life. All of these applications require tremendous computational power and with the current speed of technology development, will not be solve using a single, even the most powerful computer, a long time yet. Further decrease of computation time can be achieved mainly by application parallelization and executing them using a large number of processing units. It can be done using currently available supercomputers or parallel distributed environments [9].

On the other hand, nowadays personal computers became very popular. Most of them are connected to the computer network (Internet). Therefore, there is a tempting

* Wrocław University of Technology, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław.

possibility to utilize them to help with solving the problems mentioned above. There are a number of solutions to utilize that power, however all of them require relatively large amount of user involvement – downloading and installing of applications, taking into account its safety, etc. Unfortunately most of the users nowadays are just consumers of content that is provided in the Internet and are not aware of even a bit technical aspects such as programs installing, it means that it is possible that computation power offered by their computers is not used.

Recently Web technologies became very popular, too. One of the biggest advantages of this technology is it is accessibility – it does not require any special knowledge to use it. There are webpages that have millions of visitors daily. Therefore, there is an idea to use this technology to get access to an incredible computing power that lays in regular PCs connected by the computer network.

The chapter describes a framework that utilizes Python and new extensions for JavaScript to delegate code to be executed by web site clients and gather results on server. That provides a fully functional basic environment for employing parallel distributed computing in the WWW environment [3].

The chapter is organized as follows. Section 2 introduces an idea of distributed computing over the Internet. In section 3 the architecture of developed framework is presented, as well as an example how the framework can be used. Next two sections shows the results of the first experiments that were performed using the framework. Finally, section 6 outlines the work and discusses the further works.

2. DISTRIBUTED COMPUTING OVER THE INTERNET

Distributed computing over the Internet uses specific way of application parallelization – using multiple computers to achieve a goal of improving the capability of a system. In general a distributed system can be characterized by:

- all of its components are concurrent,
- there is neither a global synchronization source nor a global clock,
- all components are independent, therefore it is possible to deal with failures of single items.

Another characteristic of such systems is that the structure of the system may not be known in advance, in terms of the network topology, latency and even the number and specification of used nodes cannot be assumed and can dynamically change. On the other hand, that kind of systems present a number of benefits – they are cost effective, they are much more reliable and are easier to extend and manage. When such a system is well designed, adding new node should be as simple as attaching a new computer and registering it, if a machine fails, the tasks that it was supposed to perform can be rescheduled on some other device.

Currently BOINC is the most frequently used platform for distributed computing over the Internet. Most of the projects that use BOINC are nonprofit projects. Originally it was created to support the SETI@home project. The BOINC platform has about 596200 active computers (hosts) located all over the world that enable processing on average computing power over 9.2 petaFLOPS, according to statistics for March 2013 [1]. The "BOINC client" is available for different software platforms such as Windows, Unix and Mac. In 2008 BOINC has enabled usage of CUDA – NVIDIA's framework for performing calculations using graphics processing units. It speeded up computations 2 to 4 times comparing with the previous BOINC edition.

Parallel distributed applications are often classified according to how often their subtasks need to communicate with each other. An application exhibits fine-grained parallelism when the tasks must communicate very frequently. This kind of parallel application can run reasonably mainly on computers with shared and fast memory. Coarse-grained parallelism is the case when the tasks still have to communicate but much less often. These kind of applications are possible to run using a number of processes on one computer or clusters of computers connected with a fast network, using different distributed environments, for example MPI. The third kind of parallelism is called embarrassing parallelism. This term describes applications that rarely or never need to communicate during their execution. Embarrassingly parallel applications are considered the easiest to parallelization and they are well suited to internet based distributed computing.

2.1. DISTRIBUTED COMPUTING IN WWW ENVIRONMENT

As it was mentioned in introduction, world wide web technologies are incredibly popular. The simplicity of the usage and user friendliness resulted with a huge number of devices that support it. The WWW at its beginning was meant as a simple system for accessing documents stored in different locations. Since that time it evolved into a platform for running fully functional applications that undertook a competition with traditional desktop applications. One of the most important milestones was presenting by Google their implementation of JavaScript engine called V8. The engine introduced a new standard of speed and allowed to use the language for more advanced tasks than just modifying HTML documents.

JavaScript is a language that is, in its most recognized implementation, a scripting language integrated into web browsers. Its main purpose is to provide a method for dynamic modifying the content of a web page. However, because of recent progress in JS engines performance, it also can be considered as a platform to run some more advanced code on a web site visitor computer. This leads to the possibility to employ those computers to perform calculations that are logically connected together and aim the common goal, that is to introduce distributed computing in WWW platform.

This is the technology that is being presented in this chapter. Using a web browser and JavaScript for distributed computing over the Internet is not a common solution. There are only a few examples of such systems, for example distributed systems for evolutionary computing [2], [7] or Collaborative Map-Reduce [6].

3. FRAMEWORK ARCHITECTURE

The overall architecture of the framework is dictated by the HTTP protocol. As it was mentioned, the technology was meant to serve documents, therefore it distinguishes a server and clients, that communicate using request-response pattern. Fortunately nowadays all web browsers provide a way to perform asynchronous calls letting the web applications communicate with server without interrupting the user. The system consists of 3 components, a web server that provides a website of which visitors are going to perform calculations, a server running the framework that manages the application execution and website clients. The website is needed exclusively to provide visitors that will be perform the calculation. The website includes a short piece of code that enables download and execute the program. Those files are provided by the server that runs the framework and also manages the calculation state and offers an interface to access data needed for processing.

The calculations begin when a client connects to the web-server by opening a website, downloads the program and other required framework libraries. The program code is provided with hardcoded variables that allow continuing calculations from an exact state. This information is managed by server. The script executed by the client is able to communicate with the server at any time. This feature is important considering unknown script execution time, which can be interrupted at any time by client who closes the website. Communication uses AJAX [5] calls, therefore client is not interrupted while reading the page. Example programs that are presented in next sections, synchronize data after performing the requested task but this was dictated by reliable test network and trying to achieve better performance, mainly. When the task processing is completed the script continues execution by asking another task. From the server point of view there are a few aspects that have to be managed. The server has to keep a track of the computations progress to be able to serve proper tasks. Moreover, it should inspect returned data and in case of missing some results (that could be lost for example by to fast client disconnecting) resume the calculations. The server, when initialized, reads all programs installed by users, executes their initializing code and waits for clients requests. The requests basically are related to program code, data access and static data access. Data access is optional, however usually it is a substantial part of the calculation process. The server also should check if the calculation is completed. The server provides an admin page that allows to check the progress of computation.

The framework internally is based on MVC (Model-View-Controller) architecture [8]. The architecture was invented in 70s by Trygve Reenskaug for use in Smalltalk-76. It distinguishes 3 logical layers: model – that is responsible for accessing and manipulating of data, view – the part that handles user interface and finally controller – that receives user input and controls the view using data from model. This division turned to fit especially well with WWW applications, that usually store data in a database which can be wrapped by model classes and display HTML which nicely matches the definition of view. In the framework there can be distinguished 3 layers – the model, that is represented by the `Data` class, controller that is responsible for handling incoming requests and serving responses, and view that is implemented in `Code` class and provides interface for implementing user programs.

3.1. A TEMPLATE OF A PROGRAM

Programs that use the framework have to be written in Python and JavaScript. The program code should be saved in `code.py` file, in `/project/example` directory on the server. The content of an example file with some comments is presented below.

```
from project import *
from template import Template
```

The code begins with importing necessary modules from framework. Importing the `Template` class is not obligatory; however it increases the code readability.

```
class CodeExample(Code):
  def __init__(self, project):
    self.project = project
```

The main class must inherit from `Code` class. In this example it's called `CodeExample`. The constructor takes one argument – the `Project` class instance that is used to access other framework components.

```
  def getCode(self):
    data = self.project.data
    if data.result: return 'continueCalc();'
```

The most important method is `getCode()`, it is called when system assesses a client with a new task. Next, the procedure checks if the calculations are not finished by evaluating `data.result` and in case it is set already, the client receives a code *'continueCalc()'* that makes it wait for next task.

```
    data.step += 1
    code = "var step = "+data.step+"; // Data fixed at server
    for(var i = 0; i < step; ++i)
      //...
    if(solved)
      send('result='+i); // Send the result"
```

The first line increments tasks counter and after that a string that contains a JavaScript code of an algorithm is generated. As in the example, the code may have some variables hardcoded at server, to make the computation start and continue till an exact moment.

```
        tpl = Template.use('workerGeneric.js', {'code':code, 'project':'example'})
        return Response(tpl, 'application/javascript')
```

Before sending, the code has to be wrapped with functions that allow asynchronous communication with server. The last line creates a response and sets a proper MIME type.

```
    def adminPage(self):
        data = self.project.data
        params = {'result':data.result, 'step':data.step}
        return Template.use('admin.tpl', params, 'projects/'+self.project.name)
```

The next method provides an admin website that is meant to provide the status and results of calculations. The response object is not necessary every time and when there is no need to change the default MIME type, a string can also be returned.

```
class DataExample(Data):
    def __init__(self, project):
        self.project = project
        self.reset()
```

Another code that should be implemented inherits from `Data` class. It handles all the variables that the clients need.

```
    def reset(self):
        self.result = None
        self.step = 0
    def get(self, key):
        return self.result
    def set(self, request):
        self.result = request['result']
        return 'OK'
```

There are 3 following methods: `reset()` initializes the environment, `get(key)` returns server value while `set(request)` sets a variable, `key` and `request` are dictionaries what allows to provide broader context of request and offers a better flexibility.

```
def project_init(project):
    project.data = DataExample(project)
    project.code = CodeExample(project)
```

The last function is invoked by framework to initialize and associate objects with the `project`. It is mandatory for the program to be recognized by the system.

4. COMPUTATIONAL EXPERIMENTS

As it was stated the most suitable applications for developed framework are applications with embarrassingly parallelism. There are a few reasons of it. One of them is a fact that the clients are not reliable. This causes that some of scheduled tasks may never return results and therefore they have to be re-invoked. When it comes to algorithms that require data from previous computation steps all of the tasks must wait until the missing one completes. When the tasks are independent this problem does not occur. An example of such task is processing some loosely related data, such as photos or sensor data. Another problem that can appear is the availability of clients. Most of the websites nowadays are visited for short time. It is caused by a number of reasons such as generally low value of information presented by majority of web sites. All of the above are disadvantages of proposed approach, however the approach offers a potentially free computational power that can overcome all of those inconveniences.

Measuring the overall performance of the considered approach turns to be a non-trivial task. The system is a set of multiple computers connected by a network and each of the clients can present a different performance. That is because of a number of reasons; one of the most important, beside the general hardware performance and the operating system, is the browser. Different web browsers, even from the same manufacturer, implement JavaScript interpreters differently, therefore they can present significantly different performance. Another factor is different computer load in the moment of performing the test. Moreover computers are connected by a network that also can be the source of significant delays.

In general, calculation begins in moment when a client opens the website, therefore it's difficult to expect all of them to open the web page at the same moment of time and to stay long enough to finish requested computations. In fact they will join computations at random moments and stay for a random time. Of course experiments can be carried out in a uniform laboratory – with exactly the same computers connected by a fast Local Area Network. That, however, does not solve the problem of performance evaluation completely. HTTP implements client-server architecture, and there is no way that the server can start up all of the clients in the exactly same time. It is up to client to request the code and data for computations. As a consequence of above consideration the following way of execution time measurement has been used during experiments. All clients' need to wait until the server allows them to begin computations. It is implemented as repeatedly querying the server for the synchronization variable and when it changes its value to "on" they begin calculations. Then the overall execution time is calculated as the difference of the time between changing synchronization variable and the moment of receiving the last client response. Unfortunately this approach introduces an inaccuracy as well. One of the most important factors that influence the result is the time period of polling the state of synchronization variable. If it is to large it may jam the network, if is to low, it may introduce a significant delay. Because of this, the measurement accuracy of presented in the next section results is about 1 sec.

5. RESULTS OF PERFORMED EXPERIMENTS

To carry out the experiments the following testing environment has been built: as a server and web-server was used desktop computer with Debian operating system and as clients, three laptops with Debian, Windows 7 and Windows Vista operating systems have been used, respectively. Environment components have been connected by 100Mbps wired and 802.11 WiFi networks. During experiments different web browsers have been used. Results presented below are the average of at least 10 performed experiments. For experiments three different algorithms have been selected, each of them represents different type of parallelism.

The first tested algorithm renders a Mandelbrot fractal in parallel. There are 3 factors that can influence on results of experiments: *Size* – the resolution of the final set, *ResSize* – the size that is to be rendered by a single client, *MaxIters* – the number of iterations per pixel. The experiments have been performed on three, two and one computers (clients) that are labeled as cli3, cli2 and cli2 respectively.

Table 1. Execution time as a function of resolution

MaxIters	Size	ResSize	cli3 (sec)	cli2 (sec)	cli1 (sec)
10	100	5	2	3	6
10	1000	50	5	7	15

In the first experiments the dependence between the execution time and resolution has been check (table 1). The obtained results are as expected, firstly when increasing the *Size* the execution time grows, and secondly received speedup is close to the number of used clients.

Table 2. Execution time as a function of number of iterations

MaxIters	Size	ResSize	cli3 (sec)	cli2 (sec)	cli1 (sec)
10	1000	50	5	7	15
50	1000	50	23	35	55
100	1000	50	59	91	144

Next experiments have been performed with the fixed *Size* and *ResSize* values. Both values have been large enough to not to be strongly affected by data delays. The results of these experiments are presented in table 2. As it has been expected execution time grows with the number of iteration. Received speedup decreases when the number of iteration grows.

Table 3. Execution time as a function of *ResSize*

MaxIters	Size	ResSize	cli3 (sec)	cli2 (sec)
50	1000	50	23	34
50	1000	100	19	28
50	1000	200	21	29
50	1000	300	30	40

The results of the last experiments presented in the table 3 show the dependence between the execution time and the size of chunks executed by client. It can be determined that the optimal size of chunk is located about 100. This value is presumably specific to the particular network and clients. When deployed in the Internet the value should be set respectively to the time clients spend on the website and its load.

The second investigated algorithm was a naive parallel sorting algorithm. The algorithm consists of two steps – sorting small chunks of data and merging them together. Sorting was implemented by the embedded JavaScipt sort() function. The second stage is to merge sorted pieces. Obtained results of performed experiments are not satisfactory, the execution time grew up. For example for a string of size 2^{20} the execution time has been doubled when using 3 clients, from 8 seconds to about 20 seconds. Obtained results bring one conclusion: the algorithm, although theoretically parallelized, does not seem to suitable for the developed framework.

The last chosen algorithm for experiments was a brute force password cracker. It presents an embarrassing kind of parallelism. Every client receives a different prefix and by adding suffixes, checks if the result of hashing is equal to desired payload. The size of chunks to compute by a single client is given as a length of permutation of searching set. This reduced communication and should maximize the speed, makes the algorithm the most beneficial for the presenting framework. The code generates all possible strings containing letters from predefined set. The prefix is sent to client that attaches all permutation of given length. During experiments the tested length of prefix have been 3 and 4 while the searched string was 5 letter long.

Table 4. Execution time as a function of prefix size

Word	PrefixSize	cli3 (sec)	cli2 (sec)	cli1 (sec)
gajah	3	86	99	131
gajah	4	21	38	43

Results of performed experiments are presented in the table 4. The tests were conducted on computers that present different performance, this is the reason for not achieving an about double performance boost. It can also be determined that results of experiments, when the size of chunks was 4, are better.

6. CONCLUSIONS

Presented technology is an interesting alternative to the existing distributed environments providing an easy and accessible platform that can be incorporated for a range of uses. The project is in current study, the first results are very promising and show that applications with embarrassing parallelism are the most suitable for proposed way of computation. The project still requires a lot of work, first of all the platform should be more stable and secure. Another feature that is missing in the framework is an ability to dynamically find the capabilities of a client computer and provide an appropriate size of task. It would also be interesting to make experiments using a real webpage in true Internet environment. Especially, to measure the parameters of problem size and data transfer when clients are not designed but are random users. All of these will improve developed and implemented framework and make it more convenient for its users.

ACKNOWLEDGEMENTS

The research presented in this chapter has been done under international project *Large-scale calculations for heterogeneous architectures* as a part of the COST IC0805.

REFERENCES

[1] BOINC Project website, http://boinc.berkeley.edu.
[2] DUDA J., DLUBACZ W., *Distributed Evolutionary Computing System Based on Web Browsers with JavaScript*, LNCS 7782, Springer-Verlag, Berlin, 2013, 183–191.
[3] DZIK M., *Distributed Computing using JavaScript*, Master Thesis, Faculty of Computer Science and Management, Wrocław University of Technology, 2013.
[4] FLANAGAN D., *JavaScript: The Definitive Guide*, 5th Edition, O'Reilly Media, 2006.
[5] GARET J.J., *Elements of User Experience: The User-Centered Design for the Web and Beyond*, Pearson Education, 2010.
[6] GRIGORIK I., *Collaborative Map-Reduce in the Browser*, 2009, http://www.igvita.com/2009/03/03/collaborative-map-reduce-in-the-browser.
[7] MERELO J.J, GARCIA A.M., LAREDOJ.J.J, LUPION J., TRICAS F., *Browser-based distributed evolutionary computation: performance and scaling behaviour*, Proceedings of the 2007 GECCO, ACM, New York, 2007, 2851–2858.
[8] REENSKAUG T., *The Model-View-Controller (MVC) Its Past and Present*, 2003, http://heim.ifi.uio.no/trygver/themes/mvc/mvc-index.html
[9] TOP500 Supercomputers (June 2013), http://www.top500.org/lists/

*parallel machines system, tasks scheduling,
resources allocation*

Zbigniew BUCHALSKI*

AN HEURISTIC PROCEDURE OF TASKS SCHEDULING IN PARALLEL MACHINES SYSTEM

This paper present results of research on the problem of time-optimal tasks scheduling and resources allocation in parallel machines system. We consider an parallel machines system consisting of *m* parallel machines. This system can execute *n* tasks. We assume that all *n* tasks are independent and number of tasks is greater than number of machines. We also assume that is constancy of resources allocation in execution time all tasks set. For some tasks processing time function the mathematical model of this problem is formulated. Because our problem belongs to the class of *NP*-complete problems we propose an heuristic algorithm for solution this problem. Some results of executed numerical experiments for basis of proposed heuristic algorithm are presented.

1. INTRODUCTION

In many production processes there arise problems of scheduling a set of tasks on parallel machines with simultaneous resources allocation. By resources we understand arbitrary means tasks compete for. They can be of a very different nature, e.g. energy, tools, money, manpower. Tasks can have a variety of interpretation starting from machining parts in manufacturing systems up to processing information in computer systems. A structure of a set of tasks and different criteria which measure the quality of the performance of a set of tasks can be taken into account [1], [3], [8]–[14], [16], [18]–[21].

The time-optimal problem of tasks scheduling and resources allocation are intensive developing, as in [3], [5]–[7], [11], [15], [17]. The further development of the research has been connected with applications, among other things, in many production processes and in multiprocessing computer systems, as in [1], [2], [4], [22].

These tasks scheduling and resources allocation problems in parallel machines systems are very complicated problems and belongs to the class of *NP*-complete problems. There-

* Institute of Computer Engineering, Control and Robotics, Wrocław University of Technology, Wrocław, Poland, e-mail: zbigniew.buchalski@pwr.wroc.pl

fore in this paper we propose an heuristic algorithm for solving of a optimization problem. The purpose of optimization is to find such a schedule of tasks on parallel machines and such an allocation of limited nonrenewable schedule length criterion is minimized.

In the second section statement of the problem is presented. Third section contains formulation of optimization problem. In the fourth section an heuristic algorithm is given and in the fifth section results of numerical experiments on the base this heuristic algorithm are presented. Last section contains final remarks.

2. STATEMENT OF THE PROBLEM

We consider an parallel machines system (as shown in Fig. 1) having m parallel machines and let $M = \{1, 2, ..., k, ..., m\}$ be the set of these machines.. This parallel machines system can execute n independent tasks from the tasks set $J = \{1, 2, \ldots, i, ..., n\}$.

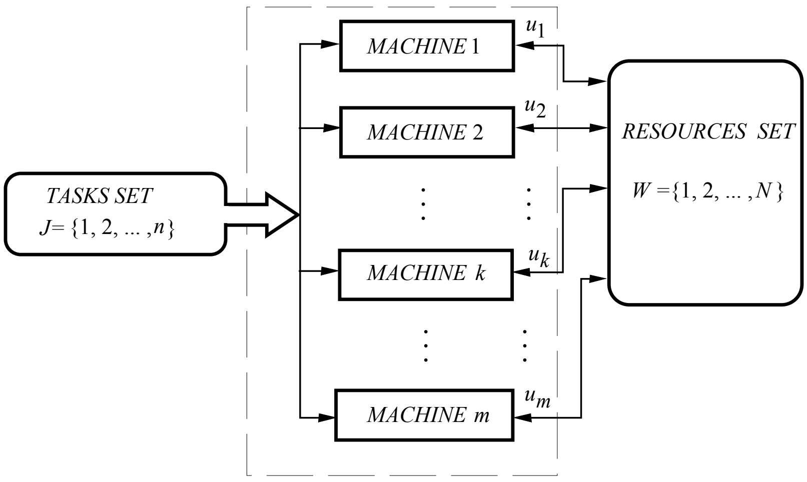


Fig. 1. Parallel machines system

We assume about this system that:

- each the machine may execute every one of n tasks,
- number of tasks is greater than number of machines ($n > m$),
- the system has N available nonrenewable resources, which are denoted as set $W = \{1, 2, ..., N\}$, $N \geq m$,
- during execution of all n tasks, the number of u_k resources is allocated to the k-th machine; $\sum_{k=1}^{m} u_k \leq N$. Each machine may use only allocated to his resources and is constancy of resources allocation in execution time all tasks set.

Results obtained in this paper are for some processing time function:

$$T_i(u_k,k) = a_{ik} + \frac{b_{ik}}{u_k}, \quad u_k \in W, \quad 1 \le k \le m, \quad i \in J, \tag{1}$$

where $a_{ik} > 0$, $b_{ik} > 0$ – parameters characterized i-th task and k-th machine, u_k is number of resources allocated to k-th machine.

This tasks scheduling and resources allocation problem in parallel machines system can be formulated as follows: find scheduling of n independent tasks on the m machines running parallel and partitioning of N resources among m machines, that schedule length criterion is minimized.

3. FORMULATION OF OPTIMIZATION PROBLEM

Let $J_1, J_2, ..., J_k, ..., J_m$ be defined as subsets of tasks, which are processing on the machines 1, 2, ..., k, ..., m. The problem is to find such subsets $J_1, J_2, ..., J_k, ..., J_m$ and such resources numbers $u_1, u_2, ..., u_k, ..., u_m$, which minimize the T_{opt} of all set J:

$$T_{opt} = \min_{\substack{J_1,J_2,...,J_m \\ u_1,u_2,...,u_m}} \max_{1\le k\le m} \left\{ \sum_{i\in J_k} T_i(u_k,k) \right\} \tag{2}$$

under the following assumptions:

$$(i) \quad J_s \cap J_t = \varnothing, \quad s,t = 1,2,...,m, \quad s \ne t, \quad \bigcup_{k=1}^{m} J_k = J,$$

$$(ii) \quad \sum_{k=1}^{m} u_k \le N, \quad u_k \in W, \quad k = 1,2,...,m,$$

$$(iii) \quad u_1,u_2,...,u_m - \text{positive integer.}$$

The assumption (iii) is causing, that the stated problem is very complicated therefore to simplify the solution our problem we assume in the sequel that the resources are continuous. The numbers of resources obtained by this approach are rounded to the integer numbers (look **Step 12** in the heuristic algorithm) and finally our problem can formulated as following minimizing problem:

$$T_{\text{opt}} = \min_{\substack{J_1,J_2,...,J_m \\ u_1,u_2,...,u_m}} \max_{1\le k\le m} \left\{ \sum_{i\in J_k} \tilde{T}_i(u_k,k) \right\} \tag{3}$$

under the following assumptions:

$$(i)\quad J_s \cap J_t = \varnothing, \quad s,t = 1,2,\ldots,m, \quad s \neq t, \quad \bigcup_{k=1}^{m} J_k = J,$$

$$(ii)\quad \sum_{k=1}^{m} u_k \leq N, \quad u_k \geq 0, \quad k = 1,2,\ldots,m,$$

where $\widetilde{T}_i : [0,N] \times \{1,2,\ldots,m\} \to R^+$ is the extension of function $T_i : \{1,2,\ldots,N\} \times \times \{1,2,\ldots,m\} \to R^+$ and formulated by function:

$$\widetilde{T}_i(u_k,k) = a_{ik} + \frac{b_{ik}}{u_k}, \quad u_k \in [0,N], \quad 1 \leq k \leq m, \quad i \in J. \tag{4}$$

Taking into account properties of the function $\widetilde{T}_i(u_k,k)$, it is easy to show the truth of the following theorem:

Theorem 1.

If the sets $u_k^*, J_k^*, k = 1,2,\ldots,m$ are a solutions of minimizing problem (3), then:

$$(i)\quad \sum_{k=1}^{m} u_k^* = N; \quad u_k^* > 0, \quad k : J_k^* \neq \varnothing, \quad k = 1,2,\ldots,m;$$

$$u_k^* = 0, \quad k : J_k^* = \varnothing, \quad k = 1,2,\ldots,m;$$

$$(ii)\quad \sum_{i \in J_k^*} \widetilde{T}_i(u_k^*,k) = \text{const}, \quad k : J_k^* \neq \varnothing, \quad k = 1,2,\ldots,m.$$

We define function $F(J_1, J_2, \ldots, J_m)$, which value is solution following system of equations:

$$\begin{cases} \sum\limits_{i \in J_k} a_{ik} + \dfrac{\sum\limits_{i \in J_k} b_{ik}}{u_k} = F(J_1, J_2, \ldots, J_m), & k : J_k \neq \varnothing, \quad k = 1,2,\ldots,m \\ \sum\limits_{k : J_{k \neq \varnothing}} u_k = N; \quad u_k > 0 & k : J_k \neq \varnothing, \quad k = 1,2,\ldots,m \end{cases} \tag{5}$$

On the basis of **Theorem 1** and (5), problem (3) will be following:

$$T_{\text{opt}} = \min_{J_1, J_2, \ldots, J_m} F(J_1, J_2, \ldots, J_m) \tag{6}$$

under the assumptions:

$$(i)\quad J_s \cap J_t = \varnothing; \quad s,t = 1,2,\ldots,m, \quad s \neq t$$

$$(ii)\quad \bigcup_{k=1}^{m} J_k = J; \quad k = 1,2,\ldots,m,$$

If $J_1^*, J_2^*, ..., J_m^*$ are solutions of problem (6), it u_k^*, J_k^* are solutions of problems (3), where:

$$u_k^* = \begin{cases} \dfrac{\sum\limits_{i \in J_k^*} b_{ik}}{F(J_1^*, J_2^*, ..., J_m^*) - \sum\limits_{i \in J_k^*} a_{ik}}; & k : J_k^* \neq \emptyset, \quad 1 \le k \le m, \\ 0 \qquad ; & k : J_k^* = \emptyset, \quad 1 \le k \le m. \end{cases} \tag{7}$$

4. THE HEURISTIC ALGORITHM

We assume that the first machine from the set *M* has highest speed and the last machine from the set *M* has least speed. We assume also if be of assistance in resources allocation so-called partition of resources coefficient γ; $\gamma > 1$. To the last *m* machine is allocated u_m resources according to the following formula:

$$u_m = \frac{N}{1 + \sum\limits_{k=1}^{m-1} [(m-k) \cdot \gamma]}. \tag{8}$$

To the remaining machines are allocated resources according to the formula:

$$u_k = (m-k) \cdot \gamma \cdot u_m; \quad k = 1, 2, ..., m-1. \tag{9}$$

The proposed heuristic algorithm is as follows:

Step 1. For given $u_k = \frac{N}{m}$ and random generate parameters a_{ik}, b_{ik} calculate the processing times of tasks $T_i(u_k, k)$ according to the formula (1).

Step 2. Schedule tasks from longest till shortest times $T_i(u_k, k)$ and formulate the list *L* of these tasks.

Step 3. Calculate mean processing time T_{mean} every machines according to follows formula:

$$T_{\text{mean}} = \frac{\sum\limits_{i=1}^{n} T_i(u_k, k)}{m}; \quad i \in J, \quad k \in M, \quad u_k = \frac{N}{m}.$$

Step 4. Assign in turn tasks from the list *L* to the first empty machine for the moment, when the sum of processing times these tasks to keep within the bounds of time T_{mean} and eliminate these tasks from the list L. Next assign to this machine task with the longest processing time from the list L so that, the sum of all tasks processing time assigned to that machine would not be greater than T_{mean} and eliminate this task from the list L.

Step 5. If there are also empty machines, which are not scheduled with tasks, go to the **Step 4**. If there is not empty machine go to the **Step 6**.

Step 6. The rest of tasks from the list L assign in proper sequence to machines according to the algorithm *LPT* (Longest Processing Time) up to moment, when the list L will be empty.

Step 7. Calculate total processing time T_{opt} of all tasks for scheduling J_1, J_2, ..., J_m, which was determined in the **Steps 3–6** and for given numbers of resources $u_k = \frac{N}{m}$.

Step 8. For given partition of resources coefficient γ allot resources u_k, k = 1, 2, ..., m to succeeding machines as calculated according formula (8) and (9).

Step 9. For tasks scheduling which was determined in **Steps 3–6** and for numbers of resources u_k, k = 1, 2, ..., m allotted to machines in the **Step 8** calculate total processing time T_{opt} of all tasks.

Step 10. Repeat the **Step 8** and **Step 9** for the next six augmentative succeeding another values of coefficient γ.

Step 11. Compare values of total processing times T_{opt} of all tasks calculated after all samples with different values of coefficient γ (**Steps 8–10**). Take this coefficient γ when total processing time T_{opt} of all tasks is shortest.

Step 12. Find the discrete numbers $\hat{u}_k$ of resources, k = 1, 2, ..., m according to follows dependence:

$$\hat{u}_{\alpha(k)} = \begin{cases} \lfloor u_{\alpha(k)} \rfloor + 1 & ; \quad k = 1,2,\ldots,\Delta \\ \lfloor u_{\alpha(k)} \rfloor & ; \quad k = \Delta+1, \Delta+2, \ldots, m \,, \end{cases}$$

where $\Delta = N - \sum_{j=1}^{m} \lfloor u_j \rfloor$ and α is permutation of elements of set M = {1, 2, ..., m} such, that: $u_{\alpha(1)} - \lfloor u_{\alpha(1)} \rfloor \geq u_{\alpha(2)} - \lfloor u_{\alpha(2)} \rfloor \geq \ldots \geq u_{\alpha(m)} - \lfloor u_{\alpha(m)} \rfloor$.

5. NUMERICAL EXPERIMENTS

On the base this heuristic algorithm were obtained results of numerical experiments for seven another values of coefficient γ = 5, 10, 15, 20, 25, 30, 35. For the definite number of tasks n = 50, 100, 150, 200, 250, 300, 350 number of machines m = 5, 10, 15, 20, 25, 30 and number of resources $N = 10.000$ were generated parameters a_{ik}, b_{ik} from the set {0.2, 0.4, ..., 9.8, 10.0}. For each combination of *n* and *m* were generated 40 instances. The results of comparative analysis of heuristic algorithm proposed in this paper and the algorithm *LPT* are showed in the Table 1.

Table 1. The results of comparative analysis of heuristic algorithm and algorithm *LPT*

n/m	Number of instances, when:			Δ^H	S^H	S^{LPT}
	$T_{\text{opt}}^H < T_{\text{opt}}^{LPT}$	$T_{\text{opt}}^H = T_{\text{opt}}^{LPT}$	$T_{\text{opt}}^H > T_{\text{opt}}^{LPT}$	%	sec	sec
1	**2**	**3**	**4**	**5**	**6**	**7**
50/5	19	2	19	2.1	2.1	1.8
100/5	20	1	19	2.6	4.2	3.7
150/5	21	0	19	3.3	7.7	6.9
200/5	23	1	16	4.5	9.4	7.9
250/5	24	2	14	5.2	11.1	9.8
300/5	25	1	14	5.9	12.4	10.5
350/5	26	0	14	6.5	14.0	10.9
50/10	20	0	20	2.4	3.7	2.9
100/10	22	1	17	3.2	5.6	4.4
150/10	23	1	16	4.1	7.9	5.8
200/10	24	2	14	4.9	9.9	6.4
250/10	24	1	15	5.7	12.2	9.7
300/10	25	1	14	6.3	14.7	10.2
350/10	26	2	12	7.5	15.8	11.9
50/15	20	1	19	1.9	4.2	2.7
100/15	20	2	18	2.1	5.6	3.9
150/15	22	2	16	3.6	6.9	5.3
200/15	23	1	16	4.7	8.2	6.5
250/15	24	2	14	5.4	10.6	8.5
300/15	25	2	13	5.9	13.8	10.9
350/15	26	1	13	7.1	16.5	13.4
50/20	19	2	19	1.7	4.4	3.6
100/20	20	1	19	2.6	6.8	5.9
150/20	22	0	18	3.4	8.7	6.9
200/20	23	1	16	4.6	10.8	8.2
250/20	24	1	15	5.7	12.6	10.6
300/20	25	0	15	6.9	14.4	12.3
350/20	26	2	12	7.8	16.7	14.7
50/25	21	1	18	1.8	5.2	4.6

1	**2**	**3**	**4**	**5**	**6**	**7**
100/25	22	2	16	2.5	7.0	5.8
150/25	23	1	16	2.9	8.7	6.8
200/25	24	1	15	4.1	10.8	8.9
250/25	25	2	13	5.3	13.6	10.7
300/25	26	1	13	5.9	16.2	13.2
350/25	27	1	12	6.9	17.4	14.8
50/30	20	1	19	2.1	4.9	3.8
100/30	21	0	19	3.2	6.8	4.2
150/30	22	2	16	3.8	8.2	5.3
200/30	24	1	15	4.6	10.4	8.9
250/30	25	2	13	5.7	13.6	11.1
300/30	26	1	13	6.6	15.4	13.2
350/30	28	2	10	8.2	17.7	15.8

In the Table 1 there are the following designations:

n	–	number of tasks,
m	–	number of machines,
T^{H}_{opt}	–	total processing time of all set of tasks J for the heuristic algorithm,
T^{LPT}_{opt}	–	total processing time of all set of tasks J for the algorithm *LPT*,
Δ^{H}	–	the mean value of the relative improvement T^{H}_{opt} in relation to T^{LPT}_{opt} :

$$\Delta^{H} = \frac{T^{LPT}_{\text{opt}} - T^{H}_{\text{opt}}}{T^{H}_{\text{opt}}} \cdot 100\% ,$$

S^{H}	–	the mean time of the numerical calculation for the heuristic algorithm,
S^{LPT}	–	the mean time of the numerical calculation for the algorithm *LPT*.

6. FINAL REMARKS

Numerical experiments presented above show, that quality of tasks scheduling in parallel machines system based on the proposed in this paper heuristic algorithm increased in compare with simple *LPT* algorithm. The few percentages improvement of time T^{H} in compare with T^{LPT} can be the reason why heuristic algorithms researches will be successfully taken in the future.

Application of presented in this paper heuristic algorithm is especially good for production systems with great number of tasks because in this case the Δ^{H} improvement is the highest. Proposed heuristic algorithm can be used not only to task scheduling in parallel machines but also to programs scheduling in multiprocessing computer systems.

REFERENCES

[1] BIANCO L., BŁAŻEWICZ J., DELL'OLMO P., DROZDOWSKI M., *Linear algorithms for preemtive scheduling of multiprocessor tasks subject to minimal lateness*, Discrete Applied Mathematics, 72, 1997, 25–46.

[2] BŁAŻEWICZ J., DROZDOWSKI M., WERRA D., WĘGLARZ J., *Scheduling independent multiprocessor tasks before deadlines*, Discrete Applied Mathematics, 65 (1–3), 1996, 81–96.

[3] BŁAŻEWICZ J., ECKER K., SCHMIDT G., WĘGLARZ J., *Scheduling in Computer and Manufacturing Systems*, Springer-Verlag, Berlin–Heidelberg, 1993.

[4] BRAH S.A., LOO L.L., *Heuristics for scheduling in a flow shop with multiple processors*, European Journal of Operational Research, Vol. 113, No. 1, 1999, 113–122.

[5] BUCHALSKI Z., *Tasks scheduling on different parallel machines with allocation of limited resources*, Zeszyty Naukowe Politechniki Śląskiej, Automatyka, No. 123, Gliwice, 1998, 77–84 (in Polish).

[6] BUCHALSKI Z., *Application of heuristic algorithm for the tasks scheduling on parallel machines to minimize the total processing time*, Proceedings of the 15th International Conference on Systems Science, Vol. 2, Wrocław, 2004.

[7] BUCHALSKI Z., *Minimising the Total Processing Time for the Tasks Scheduling on the Parallel Machines System*, Proc. of the 12th IEEE International Conference on Methods and Models in Automation and Robotics, Domek S., Kaszyński R. (Eds.), Międzyzdroje, Poland, MMAR 2006, 28–31 August 2006, 1081–1084.

[8] BUCHALSKI Z., *A Program Scheduling Heuristic Algorithm in Multiprocessing Computer System with Limited Memory Pages*, Polish Journal of Environmental Studies, Vol. 15, No. 4C, 2006, 26–29.

[9] CHENG J., KARUNO Y., KISE H., *A shifting bottleneck approach for a parallel-machine flowshop scheduling problem*, Journal of the Operational Research Society of Japan, Vol. 44, No. 2, 2001, 140–156.

[10] GUPTA J.N.D., HARIRI A.M.A., POTTS C.N., *Scheduling a two-stage hybrid flow shop with parallel machines at the first stage*, Annals of Operations Research, Vol. 69, No. 0, 1997, 171–191.

[11] HOOGEVEEN J.A., LENSTRA J.K., VELTMAN B., *Preemptive scheduling in a two-stage multiprocessor flow shop in NP-hard*, European Journal of Operational Research, Vol. 89, No. 1, 1996, 172–175.

[12] JANIAK A., KOVALYOV M., *Single machine scheduling subject to deadlines and resources dependent processing times*, European Journal of Operational Research, Vol. 94, 1996, 284–291.

[13] JÓZEFCZYK J., *Task scheduling in the complex of operation with moving executors*, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, 1996 (in Polish).

[14] JÓZEFCZYK J., *Selected Decision Making Problems in Complex Operation Systems*, Monografie Komitetu Automatyki i Robotyki PAN, t. 2, Oficyna Wydawnicza Politechniki Wrocławskiej, Warszawa–Wrocław, 2001 (in Polish).

[15] JÓZEFOWSKA J., MIKA M., RÓŻYCKI R., WALIGÓRA G., WĘGLARZ J., *Discrete-continuous scheduling to minimize maximum lateness*, Proceedings of the Fourth International Symposium on

Methods and Models in Automation and Robotics MMAR'97, Międzyzdroje, Poland, 1997, 947–952.

[16] JÓZEFOWSKA J., MIKA M., RÓŻYCKI R., WALIGÓRA G., WĘGLARZ J., *Local search meta-heuristics for discrete-continuous scheduling problems*, European Journal of Operational Research, 107, 1998, 354–370.

[17] JÓZEFOWSKA J., WĘGLARZ J., D*iscrete-continous scheduling problems – mean completion time result*, European Journal of Operational Research, Vol. 94, No. 2, 1996, 302–310.

[18] JÓZEFOWSKA J., WĘGLARZ J., *On a methodology for discrete-continous scheduling*, European Journal of Operational Research, Vol. 107, No. 2, 1998, 338–353.

[19] NG C.T., CHENG E.T.C., JANIAK A., KOVALYOV M.Y., *Group scheduling with controllable setup and processing times: minimizing total weighted completion time*, Ann. Oper. Res., 133, 2005, 163–174.

[20] NOWICKI E., SMUTNICKI C., *The flow shop with parallel machines. A Tabu search approach*, European Journal of Operational Research, 106, 1998, 226–253.

[21] SANTOS D.L., J.L., DEAL D.E., *An evaluation of sequencing heuristics in flow shops with multiple processors*, Computers and Industrial Engineering, Vol. 30, No. 4, 1996, 681–691.

[22] WĘGLARZ J., *Multiprocessor scheduling with memory allocation – a deterministic approach*, IEEE Trans. Comput., C-29, 1980, 703–710.

machine learning, load balancing,
query processing, grammar analysis, data warehouse

Michal GORAWSKI*

LOAD BALANCING IN DATA WAREHOUSE – EVOLUTION AND PERSPECTIVES

The problem of load balancing is one of the crucial features in distributed data warehouse systems. In this article original load balancing algorithms are presented. The Adaptive Load Balancing Algorithms for Queries (ALBQ) and the algorithms that use grammars and learning machines in managing the ETL process. These two algorithms base the load balancing on queries analysis, however the methods of query analysis are quite different. While ALBQ bases on calculation of computing power and available system assets, the gaSQL algorithm includes direct grammar analysis of the SQL query language and its classification using machine learning. The WINE-HYBRIS algorithm that uses the CUDA architecture and Cloud Computing will be presented as a platform for developing the gaSQL algorithm.

1. ADAPTIVE LOAD BALANCING ALGORITHM FOR QUERIES (ALBQ)

The ALBQ [15] was designed as a load balancing algorithm in IMR (Integrated Meter Reading)/DSTDW (Distributed Spatial Telemetric Data Warehouse) environment basing on research presented in [11], previous versions of load balancing algorithms used in these systems are presented in [1]–[4], [17]. It adapts system state to each processed query. It is done through monitoring, and if necessary modifying system state, before query processing [5]–[7], [9], [12], [13]. The ALBQ also allows performing the balancing action regardless of system changes. The main module in the ALBQ is the balancing process overseer module (denoted as a server), which oversees and manages nodes and all system processes. System nodes main function is still storing and processing data, however they also have possibility to communicate with other nodes, send data and transmit information about their load [8], [10]. The node's load includes its computing power as e.g. mean load of the CPU in a certain time-

* Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, ul. Baltycka 5, 44-100 Gliwice, Poland e-mail: mgorawski@iitis.pl

frame. Moreover, the algorithm expects that the node can assess (or appoint) the cost of query processing, before it decides whether to process it. To minimize the loss of data in case of a node failure, the ALBQ divides data space into regions, and assumes that the tuples existing in the same region cannot be stored only in one node. This solution is similar to using the partitioning attribute [4], [10]. There are several parameters that influence and adjusts algorithms to users' needs, the most important are **inertia parameter** (**IP**), **overload threshold** (***OT***) and **computing power** (***P***):

- The inertia parameter in the ALBQ algorithm determines modification method of load balancing results. The inertia parameter takes the value $\in < 0,1)$
- The overload threshold, determines the frequency of a balancing process activation – it is the load deviation from the reference value. The low value of this parameter implies that even the slightest load deviation will be considered as the imbalance state

The **computing power *P*** can vary while system operates and it reflects changes in the node's load. The actual value of the *P* parameter does not have to be directly connected with any node's feature, but its value has to adequately define node's computing abilities.

1.1. QUERY PROCESSING PHASES

The query directed to the data warehouse is delivered to the server. Upon arrival the query is broadcasted to registered nodes, in which the query is analyzed in terms of required computing power. The computing power needed to process the query is directly proportional to the number of tuples needed to be processed. To simplify the computing power can be described as the estimated number of tuples being processed by the query. Although the estimated value along with the current load is sent back to the server, the server load is unchanged. After collecting responses from all nodes that processed the query, server normalizes the loads with regards to the estimate increase of their value and appoints the **imbalance factor (overload)** b_i for each node. If the value of b_i is greater than the default threshold, it is sent to the node, which appoints the number of tuples that should be moved to other nodes.

This value is set in a manner that allows the node load to be within boundaries of system's mean nodes load.

$$\Delta t = \frac{b_i \times P}{100} \tag{1}$$

where Δt is a number of tuples to be transferred, b_i – imbalance factor (value of a load that needs to change in order to return to balances state), a P node's computing power.

1.2. BALANCING THE SYSTEM

In a balancing process we can distinguish two main stages. This division is enforced by the need to assure the stability of nodes that do not need balancing, while performing balancing actions in imbalanced nodes. In the first stage of load balancing the server gathers data sent from nodes, and decides the actions to be taken in order to balance the system. This is implemented in the *Balance* method, which returns total values for given node. The obtained value actually is the node's overload value that should be deducted from node and the decision how it should be realized is made by the node.

```
Balance method
Set $L_p$ = default load
Set $L_{max} = L(N_{max})=max(L(N_i))$ = max load through nodes
If $L_{max} \leq L_p + p$ terminate method
Set $L_{min} = L(N_{min})=min(L(N_i))$ = min load through nodes
Transfer adequate load value l from $N_{max}$ to $N_{min}$
Set $l=min(L_{max}-L_p, L_p-L_{min})$
Include the inertia parameter
Store the transfer in the transfer matrix $m[N_{max}][N_{min}]+=l$:
Go to begin of the method
```

Where: Li i-th node load equal $L(N_i)$; p – overload threshold; m – result matrix which is temporary division policy

While receiving imbalance factor *bi* the node calculates the number of tuples to be transferred Δt. Then the adequate number of tuples is appointed for transfer and their statistics are sent to the server. These statistics will be used in a second stage of load balancing, when the server decides to which node the tuples should be transferred. This is implemented by the *Dispatch* method, called by the server.

```
The Dispatch method
Set $k = \sum_{j=1}^{m} T(R_j)$
Set $k_{left}$ as a number of tuples to transfer = k
Set $K_i$ as a number of tuples that should be transferred to the $N_i$ node {CreateDistribution method}
For each $R_j$ region
For each destination node $N_i$
Set $t = \frac{T(R_j) \times K_i}{k}$
Store the tuples division in a result matrix: $S_{ij} = t$
Decrease the number of tuples for transfer: $k_{left} = k_{left} - t$
If $k_{left} = 0$ terminate method
```

```
For each region $R_j$
Set $r_{left}$ = number of not allocated nodes in region $R_j$
Set $N_d$ = the last destination node for tuples from $R_j$
For each node $N_i$
$N_d$ = logically next node towards $N_d$
Set $n_{left}$ = number of nodes that can be sent to $N_d$
If the number of tuples in $N_d$ is lower than $K_d$ then
$S_{dj} = min(n_{left}, r_{left})$
$r_{left} = min(n_{left}, r_{left})$
If $r_{left} = 0$ then the last destination node for tuples from $R_j = N_d$
```

1.3. DIVISION POLICIES AND EXPERIMENTS

The key element in the ALBQ is a module that stores and manages **division policies.** These policies are integral part of the algorithm and decide how the data (new or already stored) should be broadcasted through the whole system. There are two types of policies: **Main Division Policy (*MDP*)** and **Temporary Division Policy (*TDP*)**

The *MDP* decides how the tuples arriving to the system should be distributed, the total values of a *MDP* for nodes should be constant during system activity. The *TDP* is set during the Balance method and denotes how the tuples should be transferred (the *Dispatch* method). The *TDP* is created for a certain node and it presents the load that should be transferred to other nodes that exists in *TDP*. Because of the *TDP* characteristics the total values can vary. The experiments performed for 1.000.000 tuples, with 2% threshold and the inertia factor set to 0.5. Obtained results (Fig. 1, Test 3.9) are compared to the case of system without adaptive balancing (Fig. 1, Test 3.8).

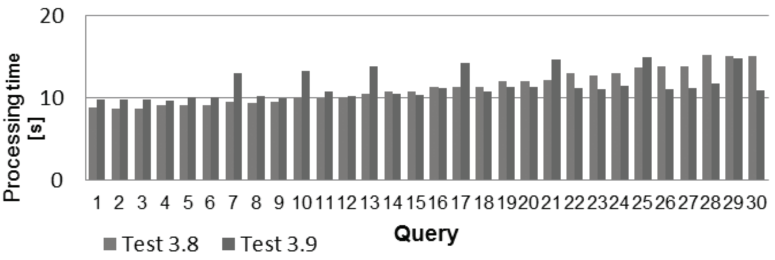


Fig. 1. Query execution time in ALBQ

We can observe more dynamic increase of processing time in a system without adaptive balancing which is caused by a larger number of processed nodes. If the node's computing power drops to 80% the adaptive balancing starts to be efficient. Moreover while computing power drops to 55% the ALBQ causes that the query processing is faster than in a system without load balancing.

2. THE WINE-HYBRIS AND THE GASQL ALGORITHM

Modern systems tend to implement the zero-latency (real-time) or near zero-latency ETL processes that updates data without interrupting the data warehouse activities [20]. The data that users obtain are most up-to-date. While designing the zero-latency ETL process there are some problems to be faced [21]. In certain solutions it is hard to assign the processing priority to users queries, and the updates generated by system [2]. The other complicated task is foreseeing the query processing time. The problems mentioned above should be taken into consideration while designing the zero-latency ETL process. To successfully cope with the real-time requirements the ETL process have to solve the problem of assigning priorities to processed queries and upcoming updates. One of the simplest method is the use of FIFO algorithm, however this leaves the system highly ineffective and does not adjusts itself to the data warehouse working environment. Other solution that can oversee the ETL process is the LEMAT system [16] based on the WINE algorithm [18]. Moreover, to meet the goals of adjusting to current system state, the LEMAT system uses the SVN learning machine. The next solution implements the load balancing problem is the original solution – the WINE-HYBRIS algorithm that uses CUDA technology [19], [27] and Cloud Computing to speed-up query responses [14].

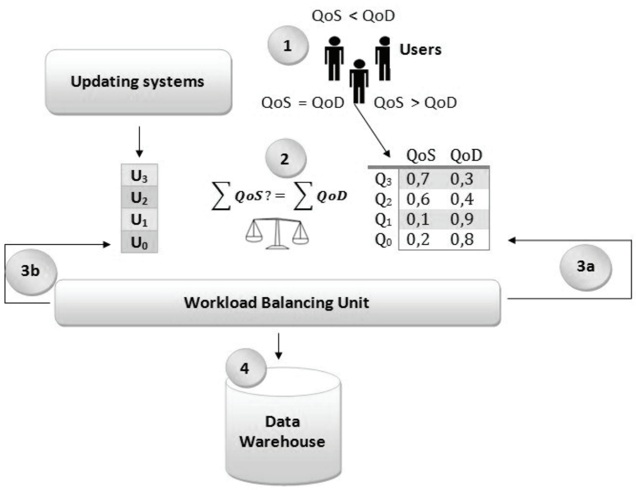


Fig. 2. The main processing scheme of the WINE-HYBRIS

The use of Workload Balancing Unit (WBU), controls all operations related to the processing and analysis of both updates and queries in the data warehouse (DW). The base that was used to create the WINE-HYBRIS algorithm is a two-level scheduling WINE algorithm (Fig. 2). Using the particular characteristics of the WINE algorithm, it was possible to incorporate user preferences, balancing and prioritization of queries and updates. Such approach enables the balancing to: a) maintain the freshness of data at the appropriate level, and b) respond to users' queries. The algorithm also uses the term partition that can be presented as a subset of the data warehouse's data table. The tests performed in the WINE-HYBRIS environment based on two architectures 1) CUDA and CPU (Fig. 3) and 2) Cloud Computing.

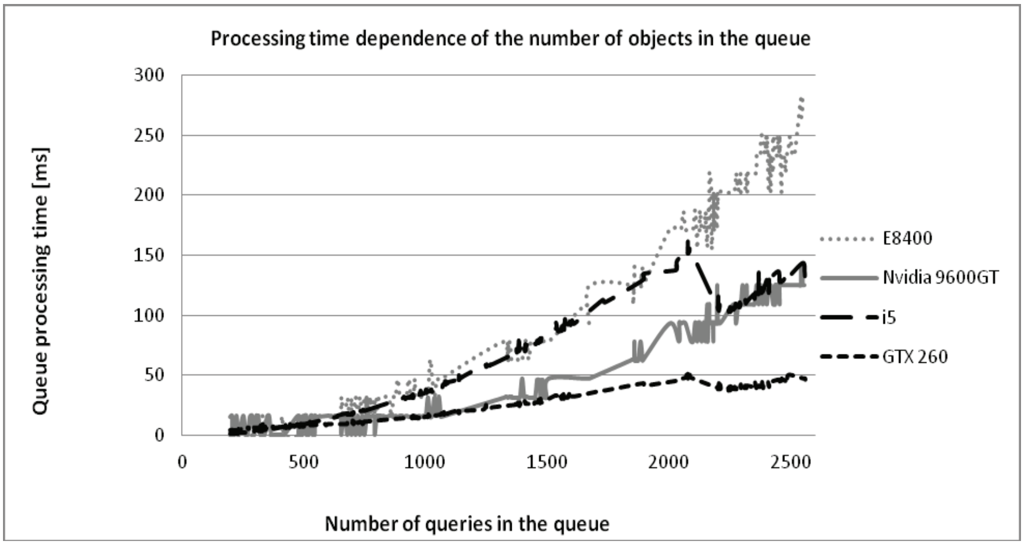


Fig. 3. Comparison of CPU and GPU architectures

All experiments were performed on two different machines, in order to verify the behavior of data extraction in different hardware environments.

Tests proven that the use of both the CUDA architecture as well as the Windows Azure platform (Cloud Computing) increases performance, which translates into more efficient data processing. An additional advantage of Cloud Computing, is the costs optimization – you only pay for the allocated capacity in a given period of time. This also relieves allocated resources while not processing any queries and updates in the data warehouse. While creating the WINE-HYBRIS algorithm, method, which is responsible for packet prioritization queries and updates, was enhanced with the schema of two algorithms with FIFO Group. It is based on the simplest solution, resulting in minimizing the number of operations on queues during processing queries and updates.

2.1. QUERY ANALYSIS IN GASQL

During implementation the usage of a Ridor classifier was proposed for machine learning. The general schema of above mentioned solutions can be described as follows: first the query content is analyzed, and then processed with machine learning, an adequate speed class is assigned to it. In other research the use of OneR learning machine was proposed along with usage of a larger number of classifiers. The system created for the ETL management bases on the researched LEMAT system. It also includes query analysis and classification and the load balancing method. The main part is terms of query processing is the module of queries analysis and classification. If properly configured it can minimize the analysis time and increase the number of properly classified queries. The query is processed in a following manner: the update or query arriving to the system activates the algorithm. If the incoming event is a query it is analyzed and classified as slow, medium or fast (in terms of processing time). Information about classification result along with certain analysis data is rerouted to the queuing module. Next actions are taken in the load balancing module. If the query is allowed to be executed, it is processed and the information about the real processing time is sent to the classification module. There the information is stored and includes in the actions of a learning machine. During the balancing process, when the update request appears it is analyzed, and the chosen information is forwarded to the load balancing module. In the LB module update is queued and executed in an appropriate moment. Updates are queued using FIFO. The LEMAT system used for query analysis bases on regular expressions. To make this mechanism more efficient algorithms basing on direct grammar analysis connected with the SQL grammar syntax was proposed. There are basic issues connected with grammar analysis that should be presented: (1) Lexical symbol, denoted as sequence of signs matched to certain pattern (2) Non-terminal symbol, denoted as the representation of a lexical symbols sequence (3) Production, denoted as a manner in which lexical symbols and non-terminal symbols connects to create signs (4) Introduction of symbols sequences (non-terminal and terminal) A and B that are a sequences of used productions that leads from A to B.

In the presented SQL query grammar analysis we need lexical analysis and syntax analysis. The lexical analysis is responsible for downloading and converting source data into lexical symbols stream that can be used in the syntax analysis. During this stage source signs are read, grouped in lexical symbols and forwarded for further analysis, moreover there can be an initial analysis of input signs. In the second stage of the analysis, the syntax analysis, the lexical symbols sequences, generated by the lexical analyzer are checked if they can be generated using grammar. The syntax analysis relies on grouping the source program lexical symbols into grammar expressions, which can be used by the compiler to synthesize the result code.

2.2. MACHINE LEARNING AND CLASSIFIERS

In a data warehouse load balancing module, queries are queued according to the WINE and the LEMAT. One of main algorithm parameters is the expected time of query processing in a data warehouse. Queries classification uses learning machines, and appropriate classification strongly influences the efficiency of a data warehouse load balancing module. To evaluate learning machines we use the efficiency parameter (*S*), which denotes the quotient of a adequately classified instances of analyzed objects (n_p), to the number of all analyzed objects (n_c):

$$S = \frac{n_p}{n_c} \tag{1}$$

In case of a presented system, the instance is the classified query. There are also defined following terms:

- All-out Efficiency (AE) – the efficiency value for all queries appearing in the system.
- Temporary Efficiency (TE) – is a temporal evaluation of classifiers efficiency. It is a value of efficiency for all queries appearing in the system from the last learning process

To increase the effectiveness of query classifiers, we implemented classifiers from the Lemat system e.g. SVM, OneR and Ridor. The Ridor classifier was also used in a WINE_HYBRIS. The OneR learning machine proven to be the most efficient during classifiers tests, provided by the WEKA library while working in the Lemat system. The idea of using multiple classifiers was bases on mechanisms used in implementing single learning machine, with an additional transitional module between classifiers and the load balancing module.

3. SUMMARY

The presented ALBQ algorithm bases mainly on the analysis presented in [10]. The algorithm balances the nodes load before processing the actual query, which minimizes the time needed to obtain query response. The test shows quite useful features of the ALBQ when used in a system with nodes incapable of constant utilization of their full computing power. The next ALBQ's useful feature is ability to divide the data in a ETL update process basing on statistical data collected during system activity. The WINE-HYBRIS along with modified query classification gaSQL proves to be quite effective when using the CUDA technology and cloud environment. The method of a grammar analysis used in gaSQL is faster and more efficient then analysis using regular expressions in the LEMAT system. Moreover, the idea of using multiple classifiers enables system to adapt to incoming queries or improving the query instances optimization.

Currently, author works on a load balancing problem and mobility models [25] in a cellular networks environment, as well as problems connected with data stream processing paradigm [26] and user identity unification [28][29] Along with an introduction of the Long Term Evolution (LTE) network technology, new challenges of solving the load balancing problem emerged.

- In Distributed Data Warehouses (DDW) main goal of load balancing is minimizing the Minimal Response Time (MRT) which is a mean time interval between query arrival time and its response time.
- In Cellular Networks (CN) main goal of load balancing is to distribute the traffic from highly loaded cells to other parts of the system thus optimizing the availability of radio resources through the whole network.

Although these two environments and load balancing goals are virtually quite different, however the above-presented algorithms could be partly adapted from the data warehouse environment to fit other distributed systems e.g. the cellular network system. The load balancing is still an unresolved issue in LTE networks, where only basic methods are implemented as a standard and each vendor freely introduces its own solutions to the system. There are several methods of dealing with overloaded cells in LTE, e.g. adjusting the pilot power. Although manually controlled larger pilot power gives more range and larger coverage area, it can lead to creating coverage holes. The other method of load balancing, more similar to load balancing in data warehouse is to modify handover region between neighboring cells so users can migrate from heavy loaded cells to less loaded ones. Such approach is called mobility load balancing (MLB) [22]–[24] and is a part of Self-Organizing Network (SON). The load in, case of data warehouse are data which have to be loaded to the DDW, and in case of CN are the mobile devices connecting to base stations. Above mentioned author load balancing algorithms could be partially adapted as a MLB in LTE networks.

The other important research perspectives of load balancing in DW, are solutions to data privacy problem [30], and data recovery [31], [32].

REFERENCES

[1] GORAWSKI M., CHECHELSKI R., *Spatial Telemetric Data Warehouse Balancing Algorithm in Oracle9i/Java Environment*, Oracle9i/Java Environment, Intelligent Information Processing and Web Mining, 2005, 357–365.

[2] GORAWSKI M., BANKOWSKI S., GORAWSKI M., *Selection of Structures with Grid Optimization, in Multiagent Data Warehouse*, IDEAL, 2010, 292–299.

[3] GORAWSKI M., GORAWSKI M., *Modified R-MVB Tree and BTV Algorithm Used in a Distributed Spatio-temporal Data Warehouse*, PPAM, 2007, 199–208.

[4] BERNARDINO J.R., FURTADO P.S., MADEIRA H.C., *Approximate Query Answering Using Data Warehouse Striping*, J. Intell. Inf. Syst., 19, 2002, 145–167.

[5] GORAWSKI M., *Architecture of Parallel Spatial Data Warehouse: Balancing Algorithm and Resumption of Data Extraction*, Proceeding of the 2005 conference on Software Engineering: Evolution and Emerging Technologies, IOS Press 2005, 49–59.

[6] KOLSI N., ABDELLATIF A., GHEDIRA K., *Data warehouse access using multi-agent system*, Distrib. Parallel Databases, 2009, 29–45.

[7] PATON N. et al, *Selection of Structures with Grid Optimization, in Multiagent Data Warehouse*, The VLDB Journal, 2009, 119–140.

[8] BERNARDINO J., MADEIRA H., *Data Warehousing and OLAP*, Improving Query Performance Using Distributed Computing 12th Conference on Advanced Information Systems Engineering, 2000.

[9] PAPADIAS D., KALNIS P., ZHANG J., TAO Y., *Efficient OLAP Operations in Spatial Data Warehouses*, Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases, Springer-Verlag, 2001, 443–459.

[10] GANESAN P., BAWA M., GARCIA-MOLINA H., *Online balancing of range-partitioned data with applications to peer-to-peer systems*, VLDB, Vol. 30, 2004, 444–455.

[11] GORAWSKI M., *Advanced Data Warehouses*, Habilitation, Studia Informatica 30(3B). Pub. House of Silesian Univ. of Technology, 2009, 390.

[12] GOUNARIS A., PATON N., FERNANDES A., SAKELLARIOU R., *Self-monitoring query execution for adaptive query processing*, Data Knowl. Eng., 51, 2004, 325–348.

[13] STOCKINGER K., WU K., SHOSHANI A., *Strategies for processing ad hoc queries on large data warehouses*, Data Warehousing and OLAP, ACM, 2002, 72–79.

[14] GORAWSKI M., LIS D., GORAWSKI M., *The use of a Cloud Computing and the CUDA architecture in zero-latency Data Warehouses*, Communications in Computer and Information Science, Computer Networks, Vol. 370, Springer 2013, 312–322.

[15] GORAWSKI M., ISAKOW J., *Algorytm adaptacyjnego balansowania obciążenia zapytań w rozproszonych przestrzenno-temporalnych hurtowniach danych*, Studia Informatica, Vol. 32, 2011, 75–88.

[16] GORAWSKI M., WARDAS R., *Równoważący obciążenia system ETL, bazujący na maszynie uczącej*, Studia Informatica, Vol. 31, No. 2A, 2010, 517–530.

[17] GORAWSKI M., GORAWSKI M., *Balanced spatio-temporal data warehouse with RMVB, STCAT and BITMAP indexes*, PARELEC, 2006, 43–48.

[18] THIELE M., FISCHER U., LEHNER W., *Partition-based Workload Scheduling in Living Data Warehouse Environments*, DOLAP'07, ACM 2007.

[19] GORAWSKI M., LIS D., *Architektura CUDA w bezopoznieniowych hurtowniach danych*, Studia Informatica, Vol. 32, 2011, 157–167.

[20] BRUCKNER R., LIST B., SCHIEFER J., *Striving towards Near Real-Time Data Integration for Data Warehouses*, Data Warehousing and Knowledge Discovery, 4th International Conference, DaWaK'02, France, LNCS, Vol. 2454, 2002, 317–326.

[21] WAAS F., WREMBEL R., FREUDENREICH T., THEILE M., KONCILIA C., FURTADO P., *On-Demand ELT Architecture for Right-Time BI: Extending the Vision*, International Journal on Data Warehousing and Mining, 9(2), 2013.

[22] LOBRINGER A., STEFAŃSKI S., JANSEN T., BALAN, *Coordinating Handover Parameter Optimization and Load Balancing in LTE Self-Optimizing Networks*, VTC Spring, 2011, 1–5.

[23] KWAN R., AMOTT R., PATERSON R., TRIVISONNO R., KUBOTA M., *On Mobility Load Balancing for LTE Systems*, VTC Fall, 2010, 1–5.

[24] SUGA J., KOJIMA Y., OKUDA M., *Centralized mobility load balancing scheme in LTE systems*, ISWCS, 2011, 306–310.

[25] GORAWSKI M., GROCHLA K., *Review of Mobility Models for Performance Evaluation of Wireless Networks*, ICMMI, 2013, in print.

[26] GORAWSKI M., GORAWSKA A., PASTERAK K., *Evaluation and Development Perspectives of Stream Data Processing Systems*, Computer Networks, Vol. 370, Springer 2013, 300–311.

[27] GORAWSKI M., LOREK M., GORAWSKA A., *CUDA Powered User-Defined Types and Aggregates*, 27th International Conference on Advanced Information Networking and Applications Workshops, 2013, 1423–1428.

[28] GORAWSKI M., CHRÓSZCZ A., GORAWSKA A., *User Identity Unification in e-commerce*, ICSS, 2013, in print
[29] GORAWSKI M., CHRÓSZCZ A., GORAWSKA A., *Customer Unification in e-commerce*, Advances in System Science and Computing, Vol. 240, 2014, 163–172.
[30] GORAWSKI M., BULARZ J., *Protecting Private Information by Data Separation in Distributed Spatial Data Warehouse*, ARES, IEEE CS, 2007, 837–844.
[31] GORAWSKI M., MARKS P., *Checkpoint-based Resumption in Data Warehouses. Software engineering techniques. Design for quality*, International Federation for Information Processing, Vol. 227, Springer, 2006, 313–323.
[32] GORAWSKI M., MARKS P., *High Efficiency of Hybrid Resumption in Distributed Data Warehouses*, DEXA'05, HADIS'05, IEEE CS, 2005, 323–327.

*WWW programming,
graph database, WebGL*

Piotr KOSZULIŃSKI*
Ziemowit NOWAK*

ASvis: WEB-BASED LARGE GRAPH SEARCH ENGINE AND VISUALIZER

This paper describes problems and their solutions which occurred during the development and implementation of the search engine and visualizer of connections between autonomous systems.

1. INTRODUCTION

The problem of searching in a graph of nodes and connections and visualizing the results of this search has been known for a long time [1] and Currently it is experiencing its heydey in context of the new technology offered by the web browsers. Advanced functions of HTML5, CSS3, and JavaScript with WebGL – API for three-dimensional animation – have become a new challenge for the programmers. An example that well illustrates the potential of new technologies is a Polish project *planetoweb.net* – animated, interactive model of the solar system.

The motivation to construct search engine and visualizer using new Web technologies were connection data between existing autonomous systems (AS Autonomous System) on the Internet, collected by the Division of Distributed Computer Systems [2]. It is estimated that the current number of active autonomous systems is about 60 thousand of the approximately 70 thousand registered. The total number of connections between them is about 120 thousand, and supported subnets – about 200 thousand. There are two types of connections that exist between autonomous systems: ascending (upstream) and downlink (downstream). In some cases, for major transportation nodes (owned by Internet service providers and telecommunication companies), the number of descending connections reaches 3 thousand.

* Institute of Informatics, Wrocław University of Technology, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland.

The application to search and visualize connections between ASes (ASvis) consists of two parts: the backend – an application running on the server, and the frontend – application running in a web browser. The back end is responsible for providing data to the frontend. Communication between frontend and backend is possible by an interface that uses software architecture standard REST (Representational State Transfer).

2. FRONTEND

The frontend application was written in JavaScript (ECMAScript version 5) and it uses HTML5 and CSS3 for the presentation. To provide more convenient use, the application works without need of reloading. Once loaded, it uses HTML5 history mechanism to create the impression of a native desktop application, without losing the convenience resulting from changing URL (when switching between "sub-pages").

To build the application, the original and unique implementation of dispatcher and visualization of graph algorithms was used. Application also uses the following external libraries:

- **Three.js** – JavaScript 3D Engine,
- **js-signals** – implementation of signals,
- **Crossroads.js** – router,
- **EJS** – template system,
- **xui.js** – light DOM library,
- Supporting packets by Mariusz Nowak:
  - **es5ext** – ES5 extension,
  - **deferred** – support for operating with asynchronous calls,
  - **modules-webmake** – wrapping CommonJS modules to version compatible with ES5.

Architecture of the frontend application as follows:

- **app.js** – main object of the application – integrating the basic components,
- **util.js i xui_extends.js** – helpers,
- **routes.js** – definitions of URL routing,
- **lib/dispatcher_adapter.js** – delegation of link-clicking and sending forms, directing them to internal routing,
- **lib/resources/** – resources of REST API,
- **lib/stores/** – "data storage" (currently two – **remote**, external source of data loaded by XHR and **local**, that is cache),
- **lib/widgets/** – implementation of the architecture based on widgets; widgets consist of the appropriate part and the view (plus external templates),
- **lib/fba.js** – implementation of forced based algorithm (algorythm using the repulsive forces to deploy vertices of the graph),

- **lib/gods_finger.js**, **lib/camera_man.js**, **lib/renderer.js** – 3D rendering, operating the camera, objects, etc.

3. BACKEND

The core of the application is written in PHP. Backend communicates with database (MySQL and NodeDB), processes the results and searches the graph. Backend consists of four packages:

- **Tonic** – a simple PHP framework that allowed convenient construction of REST-API,
- **GraphMapper** – mapper communicates with the graph database and a relational database NodeDB MySQL,
- **NodeDB** – proprietary graph database implementation,
- **MySQL** – relational database management system that complements the functionality of the graph database **NodeDB**.

GraphMapper communicates with the **NodeDB** base and converts received results in the object-oriented representation. All queries and operations on graphs are supported by this packet. The most important of its classes are **NodeDBEngine** and **ObjectsMapper**. **NodeDBEngine** queries the database and ten sends the obtained results to **ObjectsMapper** that in turn converts them into an object form. After appropriate actions for the REST resource JSON is generated, which will be returned as a result in REST API. **Node-db-driver** for PHP is used to communicate with the **NodeDB** database library.

NodeDB was implemented in the Node.js environment [3]. This is JavaScript interpreter providing low-level API to the system (e.g. of a file system, sockets, http server). It uses V8 engine by Google, which is definitely the fastest available JavaScript interpreter by far. During start-up, NodeDB base imports data from My-SQL, and it simultaneously creates their graph representation in memory and normalizes the structure.

The MySQL complements the functionality of the graph base NodeDB to support these queires, which are rapidly executed by the relational base. In this package only one class is being used: MySQLEngine. It queries the MySQL base and generates JSON which is returned as the result in REST API.

4. REST API

Communication of the frontend with backend takes place through REST API. There are two resources whose responses are consistent with the documentation:

- **GET /connections/meta/[num_for],**
- **GET /structure/graph/[node_number]/[depth].**

Node queries

- **GET /nodes/find/[number]**
  Searches AS s by number and returns them in alphabetical order.
  Parameters:
  - **number [int]** – search on the basis of LIKE value%.

  Example: **GET /node/find/345** – finds all the vertices numbers beginning with "345".
- **POST /nodes/meta**
  Provides metadata of AS s numbers provided in the "address pool".
  Parameters:
  - **numbers [str]** – example: "[1234,2345,52345,234523]".

  Example: **POST /nodes/meta**

Structure queries

- **GET /structure/graph/[node_number]/[depth]**
  Provides a AS-s graph link structure, in addition – a list of numbers found AS s sorted by the number of calls and found a list of numbers that AS-sorted according to distance from the source AS.
  Parameters:
  - **node_number [int],**
  - **depth [int].**

  Example: **POST /structure/graph/345/3** – load graph structure from vertice 345 to 3 connections in depth.
- **GET /structure/tree/[node_number]/[height]/[dir]**
  Loads the tree structure of AS with a given number, with specified height and direction.
  Parameters:
  - **node_number [int],**
  - **height [int],**
  - **dir [string**] – accepted values: "in", "out", "both".

  Example: **GET /structure/tree/306/2/out** – returns a structure of nodes, where download of data from outside the structure takes place only through the main vertex.
- **GET /structure/path/[num_start]/[num_end]/[dir]**
  Finds path between specified start and target AS looking in a predetermined direction.
  Parameters:
  - **node_start [int],**
  - **num_stop [int],**
  - **dir [string]** – accepted values: "up", "down", "both".

Example: **GET /structure/paths/306/27066/both** – returns table containing structures of nodes, which are representing the shortest found path connections “up” and “down” from start to target node.

Connection queries

- **GET /connections/meta/[num_for]**
  Searches for information on how to connect to the specified vertex.
  Parameters:
  - **num_for [int]**

  Example: **GET /connections/meta/345** – finds information about connections of vertice “345”.

5. PERFORMANCE OF APPLICATION

Source data represent the structure of interconnected autonomous systems and they are included in a relational database. Due to the nature of the data, a better solution is to use the graph database. At run-time, this database should import data from relational databases and create in memory the representation of the graph normalizing the structure.

In the first version of the application, the graph database has been implemented on the basis of written in Java OrientDB database. Unfortunately, despite a fairly good documentation, the database contains too many shortcomings to cope its task.

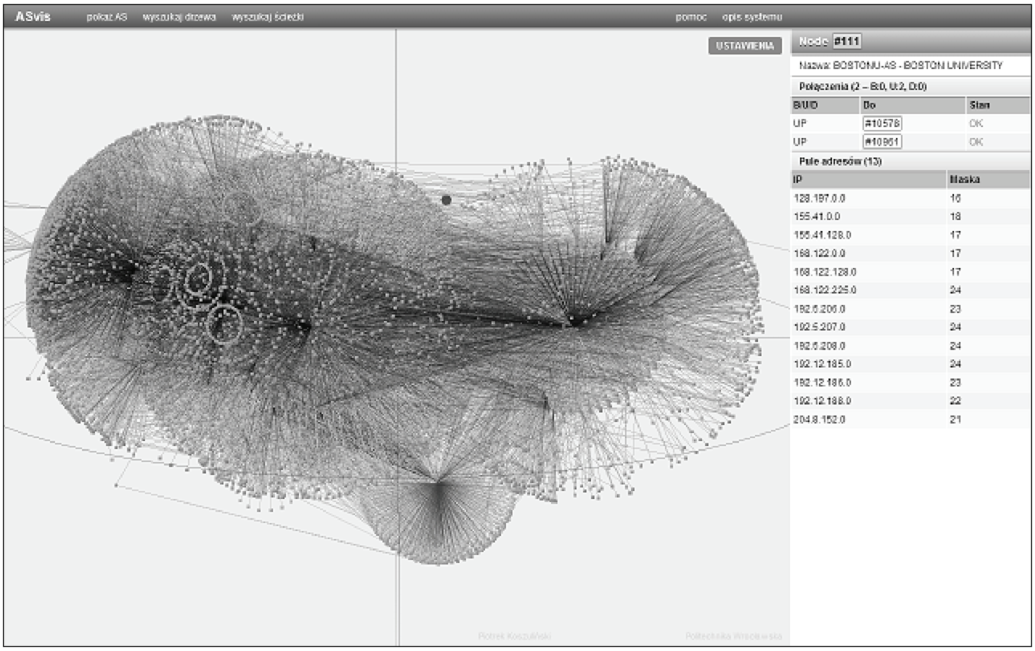
For example, downloading the structure of the search depth 4–5 lasted from 20 seconds to one minute (in extreme cases even ended in error), and it required at least 1 GB of RAM only for OrientDB base.

As a result, it was decided to implement own database of graphs, named NodeDB, in the Node.js environment. Although performance of interpreted language is generally lower, it turned out that this implementation had a surprisingly efficient performance. Time of data import from the relational database into the graph one dropped from a minimum of 2 minutes to about 5 seconds. The execution time of queries about the structure plummeted 30 times (even though it is NodeDB (not PHP) that formats output data), and the entire base is sufficient 50 MB, the memory usage has dropped 20 times.

6. WORKING WITH THE APPLICATION

To initiate the application, one should simply type in the browser the URL of the application server. This opens the page, asking you to provide a considered AS number, and the depth of the search.

The application interface is divided into two parts. On the left side a three-dimensional representation of the connections between the AS-s is displayed. The currently selected AS is marked as a big red point. On the right, information about the selected AS such as its number, name, a list of connections between ASes (along with their type and status) and a list of address fields are displayed (Fig. 1).



Node #111

Nazwa: BOSTONU-AS - BOSTON UNIVERSITY

Połączenia (2 – B:0, U:2, D:0)

B/U/D	Do	Stan
UP	#10578	OK
UP	#10961	OK

Pule adresów (13)

IP	Maska
128.197.0.0	16
155.41.0.0	18
155.41.128.0	17
168.122.0.0	17
168.122.128.0	17
168.122.225.0	24
192.5.206.0	23
192.5.207.0	24
192.5.208.0	24
192.12.185.0	24
192.12.186.0	23
192.12.188.0	22
204.8.152.0	21

Fig. 1. Search engine and visualizer interface

Navigating in a three-dimensional view has the following options:

- to rotate the graph, hold the left mouse button and drag the cursor in to a different place,
- to move the graph, hold down the Ctrl key and the left mouse button and drag the cursor to a different location,
- to zoom in/out graph, turn the scroll wheel.

When you move the cursor over any of the listed nodes, a “bubble” with the number and name of the autonomous system appears. There are a few available commands in the “bubble”: show the depth (to go to the AS-Interface and display its surrounding with the selected depth), and to show the path (this causes searching and displaying currently selected paths between currently chosen and marked ASes.)

The panel on the right side, in addition to displaying information about the selected AS allows you to navigate the connections – you can select the AS by clicking on its number on the list of connections. This behavior is a convention in the entire application: each number with the “#” prefix (eg. #12312) is a shortcut to display the considered AS with its surroundings. In addition, holding the cursor over the connection marks it on the graph with a red line.

Options

Options panel allows you to change the way in which AS-s and connections are displayed. Fog settings allow you to fade objects that are deeper in the visual field – this is useful especially for viewing large graphs. Graph settings allow you to change the size of the AS-s (green squares), and to modify the transparency of all the connection lines (up to the total transparency). This setting can turn out to be particularly indispensible in paths searching because even when displaying of the lines is turned off, line path connections are displayed. This way you can trace the path without other links obscuring visibility (Fig. 2).

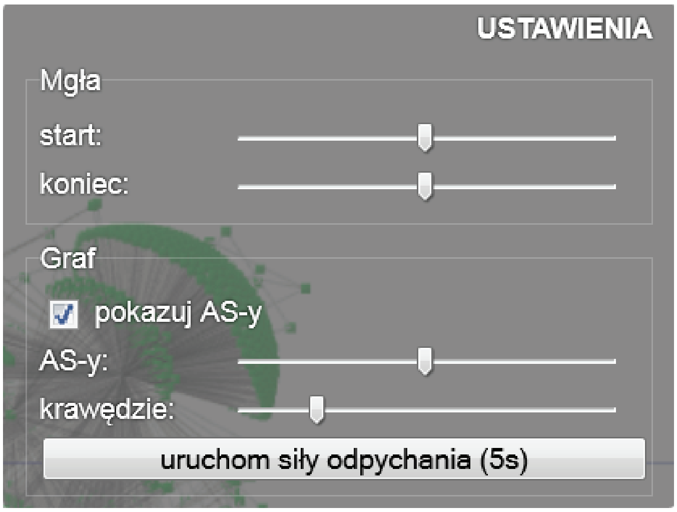


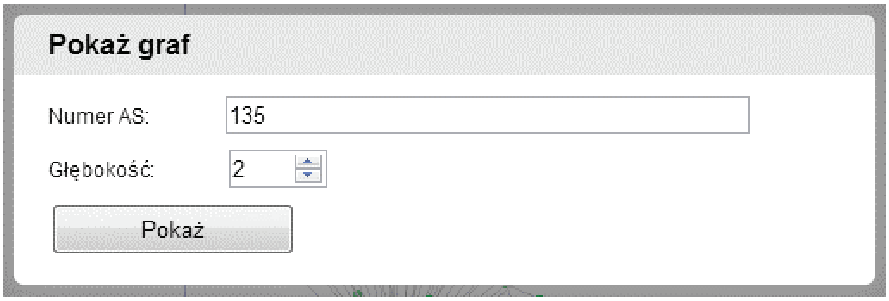
Fig. 2. Options panel

Run the repulsion button allows you to run an algorithm that improves the distribution of the graph nodes in space. This algorithm turns on automatically when you open graph connections and make them more clear by changing the distribution of nodes in the graph. This alogorithm is extremely time-consuming, so it is not executed automatically for the large graphs (over 500 AS-s). Do not use this algorithm for graphs with more than 5000 vertices, because it has a complexity of $N^2/2$.

Browsing connections

Viewing connections between AS-s consists in looking at the selected segment of the network connection. Such segment is oriented around the selected AS and is called the source. Connections are loaded according to the desired depth search.

Search Depth is a parameter that specifies how far AS-s from source can be displayed. The depth of 1, for instance, means that only the immediate neighbors can be shown – only they can be reached by going from the source with one connection; a depth of two searches for the neighbors that can be reached from the source AS with 2 connections, etc. (Fig. 3).



Pokaż graf

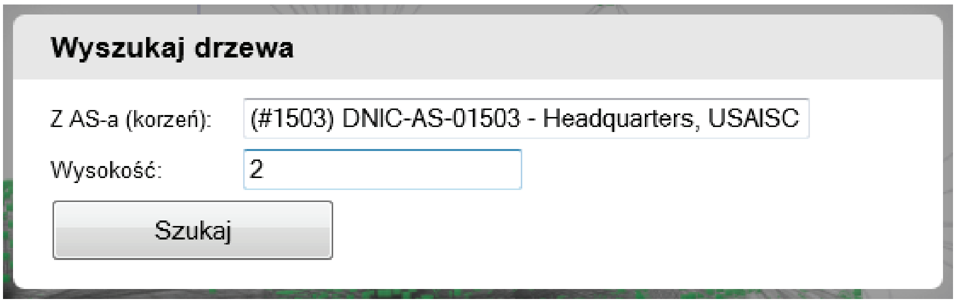
Numer AS: 135

Głębokość: 2

Fig. 3. Browsing connections

Searching of trees

The term "trees" means structures with root, but they do not have to be trees from the mathematical point of view. Such a structure should be understood as the subnet connected to the rest of the AS-s only by a root. Naturally, there can be a connection between the leaves and branches providing that they do not create an additional channel of communication beyond this subnet. To find the tree, one should select the root (the only point of communication with the rest of the world), the maximum depth of the search and the direction of connections (Fig. 4).



Wyszukaj drzewa

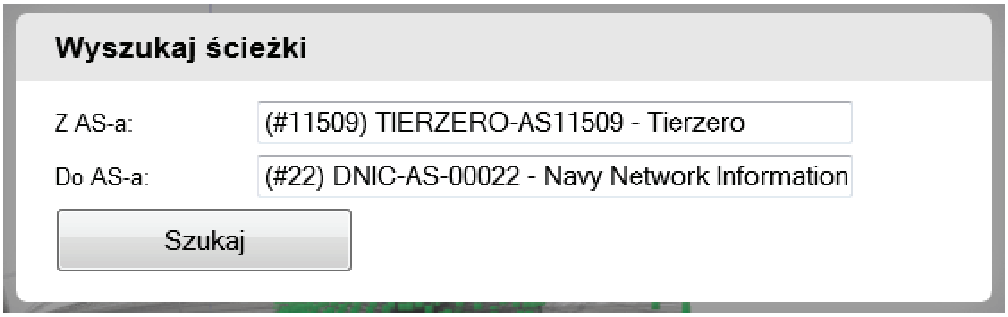
Z AS-a (korzeń): (#1503) DNIC-AS-01503 - Headquarters, USAISC

Wysokość: 2

Fig. 4. Search trees

Searching of paths

This functionality allows you to find the shortest paths connecting AS-s. Having selected a source AS, click Searching paths and enter the number of the destination AS. The shortest path (or paths if there are more than one and they are the same length) is/are highlighted in blue, (Fig. 5).


Wyszukaj ścieżki

Z AS-a:	(#11509) TIERZERO-AS11509 - Tierzero
Do AS-a:	(#22) DNIC-AS-00022 - Navy Network Information

Fig. 5. Searching of path

7. SUMMARY

ASvis application shows that the new web technologies enable advanced programming tasks, which so far have been the domain of desktop applications to be successfully done. In particular, JavaScript language is worth considering. Not only can it be executed on the client, but also on the server side; this can be indicated by the implementation of based on Node.js Node-DB graph database [3]. Analysis of communication processes between frontend and backend leads to the conclusion that a better solution would be to put graph database on frontend, however one should check whether the barrier for such a solution is the amount of memory being available to the browser.

ASvis prototype is available at *http://asvis.pwr.wroc.pl*

ACKNOWLEDGEMENTS

Authors would like to thank mgr inż. Grzegorz Kotowski, inż. Maciej Bański and Paweł Kościuk for their indispensable assistance during the implementation of the application.

REFERENCES

[1] HERMAN I., MELANCON G., MARSHAL S., *Graph visualization and navigation in information visualization: A survey*, IEEE Transactions on Visualization and Computer Graphics, Vol. 6, No. 1, 2000.

[2] KOSZULIŃSKI P., KOTOWSKI G., NOWAK Z., *Analiza struktury połączeń w Internecie na poziomie autonomicznych systemów*, Studia Informatica, Wydawnictwo Politechniki Śląskiej, Vol. 34, No. 3 (113), 2013.

[3] HUGHES-CROUCHER T., WILSON M., *Node: Up and Running*, O'Reilly, 2012.

CUDA, GPU, CPU, MPI,
K-means, computer cluster,
parallel algorithm, speedup, effectiveness

Jacek CISŁO
Dariusz KONIECZNY*

DEVELOPMENT AND TESTING PERFORMANCE OF COMPUTING CLUSTER CONSISTING OF NETWORKED GRAPHICS CARDS

The research work was designed to measure the performance of computing cluster, consisting of computers connected with network, using graphics cards to perform computations. Preparation of research environment included development of cluster, installation of environments (MPI, CUDA) and the implementation of *k*-means algorithms in versions: sequential, and parallelized (on GPU, MPI and mixture of MPI with GPU). Research consisted of measuring the execution time of the algorithms and calculation of metrics to evaluate the performance: efficiency and speedup. An influence of the data transfer on effectiveness of the algorithm parallelized on a cluster, was examined.

* Institute of Informatics, Wrocław University of Technology, Skwer Idaszewskiego 1, 50-370 Wrocław.

ISBN 978-83-7493-800-6