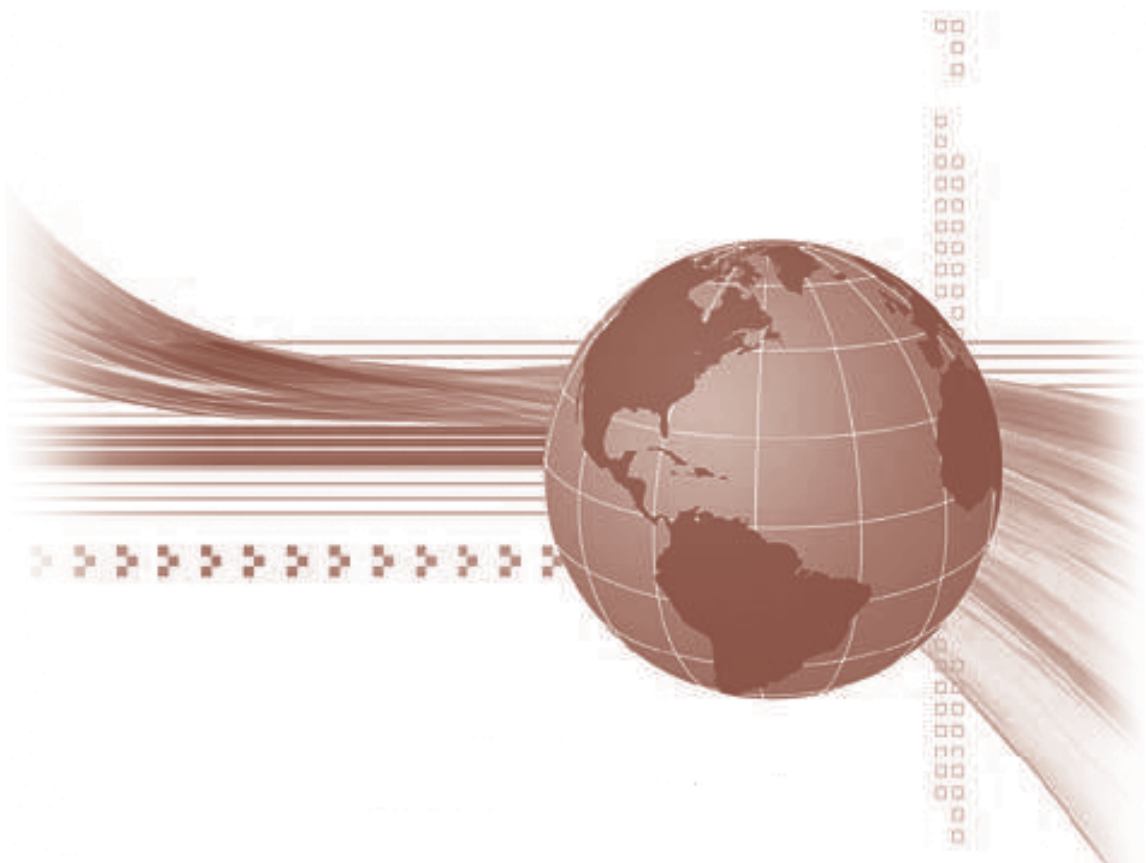
STUDIA UNIVERSITATIS
BABEŞ-BOLYAI



INFORMATICA

3/2012

YEAR **Volume 57 (LVII) 2012**
MONTH **SEPTEMBER**
ISSUE **3**

S T U D I A
UNIVERSITATIS BABEŞ-BOLYAI
INFORMATICA

3

EDITORIAL OFFICE: M. Kogălniceanu 1 • 400084 Cluj-Napoca • Tel: 0264.405300

SUMAR – CONTENTS – SOMMAIRE

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVII**, Number 3, 2012

BUILDING GRANULARITY IN HIGHLY ABSTRACT PARALLEL COMPUTATION MODELS

VIRGINIA NICULESCU

Abstract. The purpose of this study is to investigate the necessity, the existence, or the possibility to introduce mechanisms for specifying and building granularity into parallel computation models with high level of abstraction. If such mechanisms exist in this kind of models they are very useful since they allow a better evaluation of the performance, and finally an easier and more efficient implementation.

1. Introduction

One key to attaining good parallel performance is choosing the right granularity for the application. An important goal is to determine the right granularity for parallel tasks. In parallel setting the need for a unifying parallel model or a set of models is heightened by the greater demand for performance and the greater diversity among machines. Parallel computation models with high degree of abstraction usually do not have mechanisms for specifying and building granularity. If such mechanisms could be introduced they are very useful since they allow a better evaluation of the performance, and finally an easier implementation. From a practical point of view, algorithms that allow an adjustment of the granularity are preferable. It is more realistic to assume that the number of available processors is limited. A model of parallel computation, to be useful must address both issues: abstraction and effectiveness, which are summarized in the following set of requirements: abstractness, software development methodology, architecture independence, cost measures, no preferred scale of granularity, efficiently implementable [10].

We intend to focus our analysis on the models with high level of abstraction and to evaluate how the last two criteria are fulfilled. The possibility of improving this kind of models based on granularity building is going to be analyzed.

Received by the editors: June 26, 2012.

2010 *Mathematics Subject Classification.* 68Q85 , 65Y20.

1998 *CR Categories and Descriptors.* D.1.3 [**Programming Techniques**]: Concurrent Programming – *Parallel programming*; D.2.10 [**Software Engineering**]: Design – *Methodologies*.

Key words and phrases. parallel computation, models, granularity, performance, design, abstraction.

2. Level of parallelism and granularity

In modern computers, parallelism appears at various levels both in hardware and software. Levels of parallelism can also be based on the lumps of code (grain size) that can be a potential candidate for parallelism. The main categories of the code granularity for parallelism are described in Figure 1.

Grain Size	Code Item	Parallelised by
Very Fine	Instruction	Processor
Fine	Loop/Instruction block	Compiler
Medium	Standard One Page Function	Programmer
Large	Program-Separate heavyweight process	Programmer

Figure 1. The main categories of the code granularity for parallelism.

The granularity is often related to how balanced the work load is between tasks. While it is easier to balance the workload of a large number of smaller tasks, this may cause too much parallel overhead in the form of communication, synchronization, etc. Therefore, one can reduce parallel overhead by increasing the granularity - amount of work - within each task by combining smaller tasks into a single task.

Cost improvement

The size of work in a single parallel task (granularity) of a multithreaded application greatly affects its parallel performance. When decomposing an application for multithreading, one approach is to logically partition the problem into as many parallel tasks as possible. Next step is to determine the necessary communication between the parallel tasks. Since partitioning tasks, assigning the tasks to threads, and communicating (sharing) data between tasks are not free operations, one often needs to agglomerate, or combine partitions, to overcome these overheads and achieve the most efficient implementation. The agglomeration step is the process of determining the best granularity for parallel tasks [4].

If we are able to specify and to build granularity from the first levels of design, which are based on models with high degree of abstractions, then the chances to obtain good improvements of the resulted costs increase very much. This is due to the fact that the grain sizes are included in the models as parameters, and the best values for the actual parameters could be computed based on the concrete values of the target architecture.

3. Models of parallel computation

A model of parallel computation is an interface separating high-level properties from low-level ones. The role of a model is to provide a single machine - an independent, simplified target for the development and specification of programs. Unfortunately parallel computing lacks a single universally accepted model, because parallel architectures can vary drastically and because the performance of parallel algorithms depends very much on the details of the architecture. More concretely, a model is

an abstract machine providing certain operations to the programming level above and requiring implementations for each of these operations on all of the architectures below. It is designed to separate software-development concerns from effective parallel-execution concerns and provides both abstraction and stability. Since a model is just an abstract machine, models exist at many different levels of abstraction [10].

3.1. **Parallel Programming Models Classification.** Models could be classified based on their order of abstraction, in the following categories [10]:

(1): Models that abstract from parallelism completely. Such models describe only the purpose of a program and not how it is to achieve this purpose. Software developers do not even need to know if the program they build will execute in parallel. Such models are necessarily abstract and relatively simple, since programs need to be no more complex than sequential ones.

(2): Models in which parallelism is made explicit, but decomposition of programs into threads is implicit (and hence so is mapping, communication, and synchronization). In such models, software developers are aware that parallelism will be used and must have expressed the potential for it in programs, but they do not know how much parallelism will actually be applied at runtime. Such models often require programs to express the maximal parallelism present in the algorithm and then the implementation reduces that degree of parallelism to fit the designated architecture, at the same time working out the implications for mapping, communication, and synchronization.

(3): Models in which parallelism and decomposition must both be made explicit, but mapping, communication, and synchronization are implicit.

(4): Models in which parallelism, decomposition, and mapping are explicit, but communication and synchronization are implicit. .

(5): Models in which parallelism, decomposition, mapping, and communication are explicit, but synchronization is implicit.

(6): Models in which everything is explicit. Here software developers must specify all of the detail of the implementation.

Within each of these categories, we may distinguish models according to the programming paradigm on which they are based on:

-: relational/logic models;
-: functional models;
-: imperative models;
-: object-oriented models;
-: bio-inspired models.

3.2. **Models with high level of abstraction.** This category of models abstract from parallelism completely meaning that nothing related to parallelization is made explicit so the specification of the parallelism is implicit. We have different models in this category, some of them having a dynamic structure, and others using static structure as a basis. We enumerate some of them:

: *with Dynamic Structure*
  -: Higher-order Functional-Haskell, Concurrent Rewriting-OBJ, Maude

- **-:** Interleaving-Unity
- **-:** Implicit Logic Languages-PPP, AND/OR, REDUCE/OR, Opera, Palm, concurrent constraint languages
- **-:** Explicit Logic Languages-Concurrent Prolog, PARLOG, GHC, Delta-Prolog, Strand MultiLisp

: *with Static Structure*

- **-:** Algorithmic Skeletons-Cole, Darlington, P3L
- **-:** Data Parallelism Using Loops-Fortran variants, Modula 3*
- **-:** Data Parallelism on Types-pSETL, parallel sets, match and move, Gamma, PEI, APL, MOA, Nial and AT

: *with Static and Communication-Limited Structure*

- **-:** Homomorphic Skeletons-Bird-Meertens Formalism
- **-:** Cellular Processing Languages-Cellang, Carpet, CDL, Ceprol
- **-:** Data-Specific Skeletons-scan, multiprefix, paralations, dataparallel C, NESL, CamlFlight

Since these models are so highly abstract the implementer's job is very often extremely difficult; the transformation, compilation, and run-time systems must infer all of the structure of the eventual program. This means deciding:

(1) how the specified computation is to be achieved,
(2) dividing it into appropriately sized pieces for execution,
(3) mapping those pieces, and
(4) scheduling all of the communication and synchronization among them.

The second requirement is tightly connected with granularity.

Higher-order functional models. Higher-order functional models that use graph reduction for function evaluation uses parallelization for this reduction, so the granularity of the probable resulted task is not an issue of the programmer.

A similar situation is encountered for models based on *concurrent rewriting*.

Logic models. The logic models exploit the fact that the resolution process of a logic query contains many activities that can be performed in parallel. The main types of inherent parallelism in logic programs are OR parallelism and AND parallelism. OR parallelism is exploited by unifying a subgoal with the head of several clauses in parallel. Implicit parallel logic languages provide automatic decomposition of the execution tree of a logic program into a network of parallel threads. This is done by the language support both by static analysis at compile time and at run-time.

Skeletons models. The algorithmic skeletons models are based on the orchestration and synchronization of the parallel activities that are implicitly defined by the skeleton patterns. Programmers do not have to specify the synchronizations between the application's sequential parts. The communication/data access patterns are known in advance. Granularity could be an important issue for each skeleton that could be composed, but not for the entire program that just composes these skeletons.

BMF. In Bird-Meertens formalism (BMF) higher-order functions (functionals) capture, in an architecture-independent way, general idioms of parallel programming, which can be composed for representing algorithms [9, 2, 5]. Two functional are use intensively: *map* and *reduce*. The functional *map* is highly parallel since the computation of f on different elements of the list can be done independently if enough processors are available.

$$map\ f\ [x_1, x_2, \ldots, x_n] = [f\ x_1, f\ x_2, \ldots, f\ x_n] \tag{1}$$

Tree-like parallelism is expressed by the functional *red* (for reduction) with a binary associative operator $\oplus$:

$$red\ (\oplus)\ [x_1, x_2, \ldots, x_n] = x_1 \oplus x_2 \oplus \cdots \oplus x_n \tag{2}$$

Reduction can be computed on a binary tree with $\oplus$ in the nodes. The time of parallel computation depends on the depth of the tree, which is $[\log n]$ for an argument list of length n.

Other functionals may be defined, but these two are the most important and the most used.

Functions are composed in BMF by means of functional composition $\circ$, such that $(f \circ g)\ x = f\ (g\ x)$. Functional composition is associative and represents the sequential execution order.

BMF expressions - and, therefore, also the programs specified by them - can be manipulated by applying semantically sound rules of the formalism. Thus, we can employ BMF for formally reasoning about parallel programs in the design process. BMF is a framework that facilitates transformation of a program into another equivalent program. A simple example of BMF transformation is the map fusion law:

$$map\ (f \circ g) = map\ f \circ map\ g \tag{3}$$

If the sequential composition of two parallel steps on the right-hand side of Eq. (3) is implemented via a synchronization barrier, which is often the case, then the left-hand side of the equation is more efficient and should be preferred.

A very important skeleton in BMF is the homomorphism that reflects the divide-and-conquer algorithms [2, 5].

The BMF programs usually reflect the unbounded parallelism. The functional map, for example, has the time complexity equal to $O(1)$ with a processor complexity n (n is the list's length). This leads to a very fine granularity which is not practical.

In order to transform BMF programs for bounded parallelism, we may introduce the type $[\alpha]_p$ of lists of length p, and affix functions defined on such lists with the subscript p, e.g. map_p [5]. Partitioning of an arbitrary list into p sublists, called *blocks* may be done by the *distribution* function:

$$dist^{(p)} : [\alpha] \to [[\alpha]]_p \tag{4}$$

The following obvious equality relates distribution with its inverse, flattening: $red(|) \circ dist^{(p)} = id$, where $|$ means concatenation.

Using these, we obtain the following equality for the functional *map*:

$$map\ f\ = flat \circ map_p\ (map\ f) \circ dist^{(p)} \tag{5}$$

where the function $flat : [[\alpha]]_p \to [\alpha]$ transforms a list of sublists into a list by using concatenation ($flat = red(|)$).

For reduction, the transformation for bounded parallelism is as it follows:

$$red(\oplus)\ = red_p(\oplus) \circ map_p\ (red(\oplus)) \circ dist^{(p)} \tag{6}$$

We may consider that only the functions with subscription p will be distributed over the processors. The others will be sequentially computed.

PARES. For the model PARES [8] we encountered a similar approach. Functions and operators which represent programs are defined on special data structures (PowerList, ParList, PList, PowerArray, etc) based on structural induction. Two important constructor operators are used: tie ($p|q$) and zip ($p\natural q$), yielding, respectively, the concatenation and interleaving of two similar lists.

Functions are defined based on the structural induction principle. For example, the high order function *map*, which applies a scalar function to each element of a *PowerList* is defined as follows:

$$\begin{array}{l} map : (X \to Z) \times PowerList.X.n \to PowerList.Z.n \\ map.f.\,[a] = [f.a] \\ map.f.\,(p\natural q) = map.f.p \natural map.f.q \text{ or } map.f.\,(p \mid q) = map.f.p \,|\, map.f.q \end{array} \tag{7}$$

In order to increase granularity distribution functions are defined [7]; they depend on the same operator (tie or zip) that the initial non-transformed function uses.

A transformation of a function to accept distributions means that granularity could be increased for that function (program). The transformation for *PowerList* funstions is based on the following theorem.

Theorem

Given

- a function f defined on $PowerList.X.n$ as

$$f.(u|v) = \Phi^f(u, v), \tag{8}$$

  where Φ is an operator defined based on scalar functions and extended operators on *PowerList*s.
- a corresponding distribution $distr^l.p, (p \leq n)$, and
- a function f^p – *the bounded parallel function* defined as

$$\begin{array}{l} f^p.(u|v) = \Phi^{f^p}(u, v) \\ f^p.[l] = [f^s.l] \\ f^s.u = f.u \end{array} \tag{9}$$

then the following equality is true

$$f = flat_1 \circ f^p \circ dist^l.p \tag{10}$$

A similar theorem is proven for cyclic distributions, based on $\natural$ operator too. The proof could be find in [7].

In BMF of lists the granularity analysis is done only for homomorphisms. Bounded parallelism is discussed there only for concatenation operator (so only for linear distribution or block). The advantage of the PARES model is that we may use not only the distributions based on concatenation operator, but also the cyclic distributions, or combinations of distributions.

UNITY. UNITY (unbounded nondeterministic iterative transformations) [3] is both a computational model and a proof system. UNITY model uses Interleaving approach that derives from multiprogramming ideas in operating systems via models of concurrency such as transition systems. If a computation can be expressed as a set of subcomputations that commute, that means, it can be evaluated in any order and repeatedly, then there is considerable freedom for the implementing system to decide on the actual structure of the executing computation. It can be quite hard to express a computation in this form, but it is made considerably easier by allowing each piece of the computation to be protected by a guard, that is, a Boolean-valued expression. Informally speaking, the semantics of a program in this form is that all the guards are evaluated and one or more subprograms whose guards are true are then executed. When they have completed, the whole process begins again. Guards could determine the whole sequence of the computation, even sequentializing it by having guards of the form step = i, but the intent of the model is rather to use the weakest guards.

A UNITY program consists of declaration of variables, specification of their initial values, and a set of multiple-assignment statements. In each step of execution, some assignment statement is selected nondeterministically and executed. For example, the following program,

$$\begin{array}{l}\text{Program P}\\ \text{ initially } x = 0\\ \text{ assign } x := a(x) [] x := b(x) [] x := c(x)\\ \text{end } \{\text{P}\}\end{array}$$

consists of three assignments that are selected non-deterministically and executed. The selection procedure obeys a fairness rule: every assignment is executed infinitely often.

In order to initialize a matrix - $U(n \times n)$ - to unity matrix; three variants could be used:

$$\begin{array}{l}< [] \ i,j : 0 \le i < n \wedge 0 \le j < n :\\ \quad U(i,j) := 0 \ if \ i \ne j \ \sim \ 1 \ if \ i = j\\ >\end{array}$$

or

$$\begin{array}{l}< \| \ i,j : 0 \le i < n \wedge 0 \le j < n : U(i,j) := 0 >\\ [] \ < \| \ i : 0 \le i < n : U(i,i) := 1 >\end{array}$$

or

$$
\begin{array}{l}
< [] \ i : 0 \leq i < n : U(i,i) := 1 \\
[] \ < \| \ j : 0 \leq j < n \wedge i \neq j : U(i,j) := 0 > \\
>
\end{array}
$$

The first solution has n^2 statements, one for each matrix element; the second solution has two statements, and the third has n statements, one for each matrix row.

A mapping of a UNITY program expresses a way in which that program can be executed on an architecture; such a mapping may transform the connector [] into sequential or parallel composition [3].

A possible mapping onto a synchronous multiprocessor architecture that we may define is based on the following steps:

: choose one statement based on a certain order;
: execute the components of a statement (multiple assignments); the componets are executed in parallel by using the set of processors that are synchronized before other statement starts.

In this case, the connector [] means sequential composition, and the granularity analysis of the variants that initialize the unity matrix is the following:

: the first solution has the biggest granularity (nothing is executed in parallel);
: the second variant has fine granularity (two statements: the first with n^2 components and the second with n components);
: the third has a medium granularity (n statements each with two sub-statements, and the second sub-statement has n components).

If a mapping on an asynchronous architecture is considered then we have the following steps:

: each statement is allocated to a processor, and
: a sequence of execution (control flow) is defined for each processor (if more than one statement is allocated to a processor).

Statements allocated to different processors are executed in parallel. The components of a statement could be executed in parallel by using a set of processes that are synchronized before other statement starts. If we are able to transform the program in such a way that the number of statements is equal to the number of processors, then the granularity is perfect for the target architecture.

As we may see from the last variant for matrix initialization, a statement could be defined as a multiple assignment or as a composed statement (being composed from two or more sub-statements).

The granularity of a UNITY program can be adjusted by changing the way in which the assignment components are organized in the same or in different assignment statements. An important requirement is that a transformation like this has to preserve the correctness.

So, we may conclude that even if this model is so highly abstract we may change the granularity by interchanging the separation symbol [] with the symbol $\|$, by following some rules that preserve the result.

4. Conclusions

A programming model provides a set of rules or relationships that defines the meaning of a set of programming abstractions. These abstractions are manifested in a programming language which is an instantiation of that programming model. A primary objective is to allow reasoning about program meaning and correctness. Still, since the main reason for using parallelism is achieving performance, cost evaluation is also very important for a parallel computation model. Granularity is a factor that could influence the cost in a great measure. The importance of building granularity from the first stages of parallel programs development justifies the introduction of some mechanisms for specifying granularity even in models with high degree of abstraction.

For serial algorithms, the time complexity is expressed as a function of n – the problem size. The time complexity of a parallel algorithm depends on the type of computational model being used as well as on the number of available processors. Therefore, when giving the time complexity of a parallel algorithm it is important to give the maximum number of processors used by the algorithm as a function of the problem size. This is referred to as the algorithm's *processor complexity*.

The synthesis and analysis of a parallel algorithm can be carried out under the assumption that the computational model consists of p processors only, where $p \geq 1$ is a fixed integer. This is referred to as *bounded parallelism*. In contrast, *unbounded parallelism* refers to the situation in which it is assumed that we have at our disposal an unlimited number of processors. But the next question would be "how we may distribute the work across these processors". The answer is easy if the model allows different granularity scales.

Parallel programs specified using a computational model with high degree of abstraction usually reflect the unbounded parallelism, and so usually we have very fine size of granularity. But, some of these models could be adjusted to accept any scale of granularity. This represents an important advantage since based on this a more accurate cost model could be defined, too.

From the above analysis we may conclude that for the models with high degree of abstraction, which are based on domain decomposition we may define more easily formal methods for building granularity. Another important thing is that these methods have to be formally defined in order to assure the correctness preservation; the main reason of using models is to assure a software development method that assures the correctness, since debugging is extremely difficult in parallel setting.

References

[1] R. Bird. *Lectures on Constructive Functional Programming*. In M. Broy editor, Constructive Methods in Computing Science, NATO ASI Series F: Computer and Systems Sciences, Vol. 55, pg. 151-216. Springer-Verlag, 1988.

[2] M. Cole. *Parallel Programming with List Homomorphisms*. Parallel Processing Letters, 5(2):191-204, 1994.

[3] Chandy, K.M. AND Misra, J. *Parallel Program Design: A Foundation*. Addison-Wesley, Reading, MA.1988.

[4] Foster, I. *Designing and Building Parallel Programs*, Addison-Wesley 1995.

[5] Gorlatch, S., *Abstraction and Performance in the Design of Parallel Programs*, CMPP'98 First International Workshop on Constructive Methods for Parallel Programming, 1998.
[6] Misra, J.: PowerList: A structure for parallel recursion. ACM Transactions on Programming Languages and Systems, Vol. 16 No.6 (1994) 1737-1767.
[7] Niculescu V., *Data Distributions in PowerList Theory*. Lecture Notes of Computer Science, Vol. 3722: Theoretical Aspects of Computing, Proceedings of ICTAC 2007, Springer-Verlag, 2007: 396-409.
[8] Niculescu, V. *PARES - A Model for Parallel Recursive Programs*, Romanian Journal of Information Science and Technology (ROMJIST), Ed. Academiei Romane, Volume 14(2011), No. 2, pp. 159-182, 2011.
[9] D.B. Skillicorn. *Foundations of Parallel Programming*. Cambridge University Press, 1994.
[10] Skillicorn, D.B. and Talia, D.: Models and Languages for Parallel Computation. ACM Computer surveys, 30(2): 123-136, June 1998.

Department of Computer Science, Babeş-Bolyai University, Cluj-Napoca
E-mail address: vniculescu@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVII**, Number 3, 2012

UNIFORM SOLUTIONS FOR WEB SERVICES

FLORIAN BOIAN AND BEATA JANCSO

Abstract. Web Services became widely used in today's software environment. Almost every distributed software application needs at least one Web Service to enhance its functionality. A distributed application can be easily created by using the WSWrapper component. WSWrapper offers a unique and uniform solution for Web Service implementation and integration. An important component of WSWrapper called WSGenerator is presented. WSGenerator provides a simple and easy to use solution for Web Service proxy generation. The main advantage of this component is the uniform and platform independent interface. For case studies a simple Web Service named HugeIndexOfFiles is defined. This service provides interfaces for the major Web Service types: XML-RPC, SOAP and REST. For each variant the service is described using the standards: XRDL, WSDL and WADL.

1. Introduction

The Web Service technology becomes an important component of todays distributed software environment. This technology is so widely used because of its ability to realize platform independent communication between the components of a distributed application. Another important aspect to mention is the fact that a Web Service can return the requested information in several standardized formats (*XML, JSON, HTML*) which can be consumed by different type of clients, such as: browser based clients, rich desktop applications, and other business applications running on smart portable devices.

There are a couple of available solutions for Web Services. The majority of these frameworks is platform dependent, and do not offer a unique and uniform solution for all three major Web Service types: ***RPC [15]***, ***SOAP [12]*** and ***REST [10]***. Because of the lack of a uniform Web Service solution,

Received by the editors: August 17, 2012.

2010 *Mathematics Subject Classification.* 68U35, 68M11, 68N30, 68N25.

1998 *CR Categories and Descriptors.* H.3.5 [**Information Storage and Retrieval**]: Online Information Services – *Web-based services*.

Key words and phrases. Web Service, WSWrapper, WSGenerator, Formal description of interfaces.

a new solution called ***WSWrapper*** was created and presented in the papers [1, 2, 3, 4, and 8].

The ***WSWrapper*** framework offers a unique and uniform solution for creating Web Services for all the major Web Service types: ***XML-RPC***, ***SOAP*** and ***REST***. All in all, the system provides [1, 2] a unique set of objects, regardless of the implementation language or customer service. In order to implement a service, a user must perform the following actions:

- provide a set of classes, functions, or methods that is exported from the service to be executed by the clients
- define an object ***WebService [2]*** passing the service address, name and type (where type can be XML-RPC, SOAP or REST)
- define mappings for the service methods
- define actions to deploy the service

In order to implement a Web Service client, the user must perform the following actions:

- define a ***WebServiceClient [2]*** object passing the service address, name and type (where type can XML-RPC, SOAP or REST)
- invoke the "***call***" method passing the method to call, and the list of parameters

WSWrapper was implemented in the following programming languages: ***Java***, ***Python***, ***PHP*** and ***C #***.

The WSWrapper framework also offers a uniform solution for Web Service proxy generation. This new component it is called ***WSGenerator [5]*** and was added recently to the WSWrapper framework. Having a uniform Web Service proxy generator library is usefully especially for applications which use functionality supplied by several services. Using several libraries for proxy generation is not ideal, and maintaining such an application could become a nightmare. Also creating the proxy from scratch for each of the used Web Services is not the best solution. ***WSGenerator*** was introduced because of the lack of a uniform and platform independent, automated proxy generation library. The intention of ***WSGenerator*** is to simplify the development phase of a distributed application.

2. The WSGenerator component

The proxy generation component was added to the ***WSWrapper*** framework with the intention to create a complete, platform independent, and uniform set of solutions for Web Services. This new component is called ***Web Service Proxy Generator*** or ***WSGenerator***.

WSGenerator provides to the end user a simple and easy to use tool for generating client Web Service proxies. The proxy generation library offers a unified set of solution *[3]* for all the three major Web Service types: RPC, SOAP and REST. By using this component, generating a proxy for a given service becomes an easy process. The Web Service specific details and operations, including specific transformations remain hidden to the end-user.

At the moment WSGenerator provides solutions for generating client proxies in the following popular programming languages: ***Java***, ***C#***, ***Python*** and ***PHP***. In the future, support for other programming languages will be added.

The ***WSGenerator*** component can be used for generating a client proxy for any Web Service, as long as the Web Service has a machine processable service description file published. The service description file format depends on the type of the Web Service. WSGenerator uses the service description file formats shown in Table 1 for proxy generation.

Web Service type	RPC	SOAP	REST
Format	WSDL XRDL	WSDL	WADL WSDL 2.0 HTTP binding extension

TABLE 1. Web Service description file formats

In practice not all the Web Services publish service description files. Especially Web Services of types RPC and REST do not have any description file. As a consequence generating a proxy for RPC and REST services becomes a real challenge. If the service description file is missing automatic proxy generation becomes much more complicated than the manual proxy creation.

Generating a proxy for a given Web Service is a simple process and can be characterized by the following steps:

- ***step1***: the end-user requests a proxy for service X
- ***step2***: ***WSGenerator*** generates a proxy for service X and returns it to the end-user. The end user receives an executable file containing the generated proxy. For example if the end user request the generation of a proxy for a Java service, he will receive a .jar file.

WSGenerator performs the following actions for proxy generation:

- connects to service X
- reads the service description file
- generates the proxy for service X
- returns the generated proxy to the end user

If there was any error during the proxy generation process, a corresponding exception will be returned to the end-user.

At an abstract level the ***WSGenerator*** component can be defined by the interface shown in Figure 1.

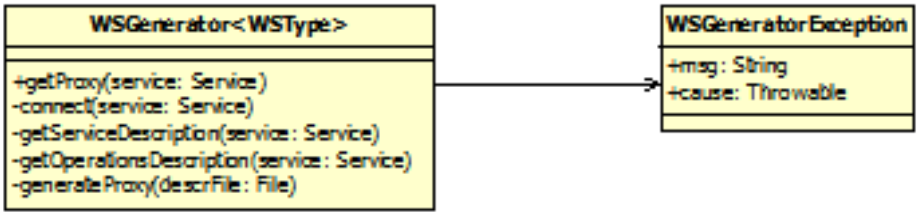


FIGURE 1. WSGenerator interface

The domain objects used by the ***WSGenerator*** interface are represented in the diagram from Figure 2.

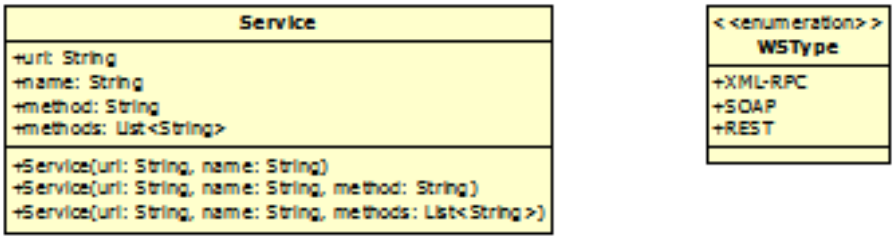


FIGURE 2. WSGenerator domain objects

At the moment the WSGenerator component uses only two domain objects: ***Service*** and ***WSType***. The ***Service*** interface represents at an abstract level a Web Service, and the ***WSType*** represents the type of the Web Service. Currently the following three major Web Service types are supported: ***XML-RPC***, ***SOAP*** and ***REST***. Besides these three Web Service types,

support for other types will be added in the future.

In some cases a Web Service might require some sort of authentication from the client application. Therefore support for authentication will be added to the next version of WsGenerator interface.

Using the ***WSGenerator*** an end-user can also generate a proxy for only one or a few methods of a given Web Service. This feature can be useful when

the client application uses only one or a few methods of a given Web Service. In such cases it makes more sense to include only the necessary methods in the proxy, instead of including all public methods of the service.

3. Case study: HugeIndexOfFiles web services

In order to test the qualities and observe the possible issues of the ***WSWrapper*** framework and ***WSGenerator*** component a web service called HugeIndexOfFiles was created.

The ***HugeIndexOfFiles*** web service indexes in databases some characteristics of the files from various directories, archives, and hosts. The number of indexed files can be very large, until tens of millions.

After the indexing has finished several information can be obtained. As an example we mention the following:

- files whose name get a check pattern,
- duplicated files (possibly with different names)

The indexing database has three tables: head, files and archives, presented in Figure 3.

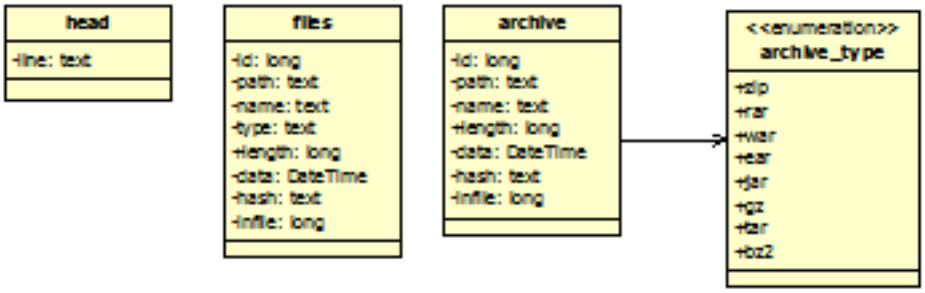


Figure 3. HugeIndexOfFiles indexing database tables

The fields of the indexing database have the following meaning:

- ***line***: line contains a statistical summary of the database content like:

Sat May 19 21:28:09 2012, db: flocopii.db, files: 412784, differentFiles: 221151, filesInArchives: 5961658, hiddenFiles: 10958, roots: +f:/

- ***id***: represents the primary key of the line from the table
- ***path:*** contains the URL of hosting machine and the path in the file system of the file,
- ***name:*** is the name of the file,

- ***type:*** represents the extension of the name (like .c, .pdf, .txt etc.),
- ***length:*** represents the size of the file in bytes,
- ***data:*** represents the creation date and time,
- ***hash***: is a SHA1 hash of the contents of the file, in base64 representation,
- ***infile***: is 0 if the file is not member of any archive file, or is the id of the archive file member. If the file is member of more nested archives, then infile will point to the largest from the archives member.

The public interface of the ***HugeIndexOfFiles*** is illustrated in Figure 4.

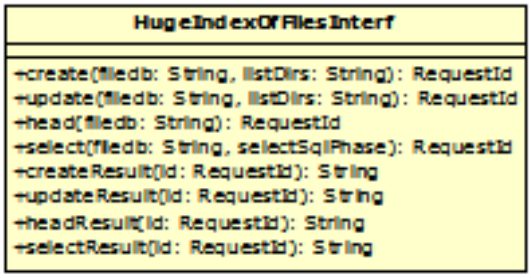


FIGURE 4. HugeIndexOfFiles interface

In some cases the indexing process can take several hours or even more days. As a consequence the ***HugeIndexOfFiles*** has four pairs of methods: one for the request, and one for the response. This policy was adopted in order to obtain the service response asynchronously. In this way, the clients will not be blocked until the service finished the indexing process. The service maintains a pool of requests and a corresponding pool of responses. A request-response pair will be removed only after the client had obtained the final response.

As observed in Figure 4 the ***HugeIndexOfFiles*** interface methods uses the following arguments:

- ***RequestId***: is a long integer that uniquely identifies a request, it is similar to the ***serialVersionUID*** used in the case of distributed applications,
- ***filedb***: contains the URL and absolute name of database,
- ***listDirs***: contains a list of directories and has the form: *DIR1, DIR2,. . . , DIRn*. A directory has the following form: ***{+|0|-} URL Absolute-path.*** The ***+*** prefix is used in order to find files in the archives. The ***0*** prefix is used to ignore the archives, and the − prefix is used to remove files from a directory in the index.

- ***selectSqlPhrase***: contains a valid SQL query

The main pair of methods of the HugeIndexOfFiles is: ***select*** and ***selectResult***. The ***head*** method checks the database consistency. If the database is in an inconsistent state, the ***head*** method transforms it so that the loss to be minimal.

For creating a new database for a list of directories the ***create*** method can be used. If the ***createResult*** or ***updateResult*** methods are invoked before the ***create*** or ***update*** operations had finished, the service response will be something like this:

```
6360000 files handled Sat May 19 21:14:54 2012
f:/Sticuri/BFMntfs2Go/01Compendiu.zip/01Compendiu/Echo.class
```

An example of a final response from ***createResponse*** and ***updateResponse*** is shown in Table 2.

createResponse	***updateResponse***
412784 inserted files. 5961658 inserted files from archives. 0 ignored files from -DIR's. Sat May 19 21:28:09 2012, db: flocopii.db, files: 412784, differentFiles: 221151, filesInArchives: 5961658, hiddenFiles: 10958, roots: +f:/ create stop. Time 613139 seconds.	186354 intact files. 0 intact files from archives. 5 created files. 18 created files from archives. 5 modified files. 7 deleted files. 0 deleted files from archives. 0 ignored files from -DIR's. Sat Apr 28 15:08:06 2012, db: rlfdata.db, files: 186364, differentFiles: 99058, filesInArchives: 3522260, hiddenFiles: 1351, roots: +d:/ update stop. Time 3426 seconds.

TABLE 2. createResponse and updateResponse response example

The main components of the ***HugeIndexOfFiles*** web service are implemented in the Python programming language. In order to test the qualities and defects of the ***WSWrapper*** and ***WSGenerator*** three implementation of the ***HugeIndexOfFiles*** were created using: ***XML-RPC***, ***SOAP*** and ***REST***. Because a ***SOAP*** service will create by default a ***WSDL [13]*** document, in the following paragraphs only the ***XRDL [16]*** and ***WADL [7]*** documents of the ***HugeIndexOfFiles*** web service implementations are presented.

In the case of the ***XML-RPC*** web service an XRDL document is created. The XRDL file of the HugeIndexOfFiles is illustrated in Figure 5.

```
<?xml version="1.0" encoding="UTF-8"?>
<service name="HugeIndexOfFiles XML-RPC service"
    ns="ro.ubbcluj.cs.hugeindexoffiles" url="www.scs.ubbcluj.ro">
    <types>
        <type name="RequestId" type="long" />
    </types>
    <methods>
        <method name="create" result="RequestId">
            <param type="string">filedb</param>
            <param type="string">listDirs</param>
        </method>
        <method name="update" result="RequestId">
            <param type="string">filedb</param>
            <param type="string" mandatory="false">listDirs</param>
        </method>
        <method name="head" result="string">
            <param type="string">filedb</param>
        </method>
        <method name="select" result="string">
            <param type="string">filedb</param>
            <param type="string">selectSqlPhrase</param>
        </method>
        <method name="createResponse" result="string">
            <param type="RequestId">id</param>
        </method>
        <method name="updateResponse" result="string">
            <param type="RequestId">id</param>
        </method>
        <method name="headResponse" result="string">
            <param type="RequestId">id</param>
        </method>
        <method name="selectResponse" result="string">
            <param type="RequestId">id</param>
        </method>
    </methods>
</service>
```

FIGURE 5. HugeIndexOfFiles XRDL document

In the case of the ***RESTful HugeIndexOfFiles*** web service the parameters ***filedb*** and ***id*** will be the last fields from the URI. The methods of the service can be invoked as presented in the table shown in Table 3.

URI	HTTP method	HTTP body
http://address/create/{filedb}	POST	listDirs
http://address/update/{filedb}	PUT	listDirs?
http://address/head/{filedb}	GET	-
http://address/select/{filedb}	PUT	selectSqlPhrase
http://address/createResponse/{id}	DELETE	-
http://address/updateResponse/{id}	DELETE	-
http://address/headResponse/{id}	DELETE	-
http://address/selectResponse/{id}	DELETE	-

TABLE 3. HugeIndexOfFiles RESTful web service method invocation

The operations of the RESTful ***HugeIndexOfFiles*** web service are described using a ***WADL [9]*** document. An excerpt from the ***WADL*** document is presented in Figure 6.

4. Conclusions

The majority of the popular distributed software application uses one or more Web Service supplied functionality. Manual creation of several client proxies is not ideal. Also using several different libraries for automatic proxy generation might not be a good solution.

The ***WSGenerator*** component offers a simple, easy to use, and unique tool for Web client proxy generation. WSGenerator becomes a unique tool because of its uniform interface and support for the most popular Web Service types. The client proxy is created based on the service descriptor file of the Web Service. If the service does not publish any service description file, automatic proxy generation might be difficult or even impossible. This might be an issue especially in the case of ***RPC*** and ***RESTful*** Web Services. As a possible solution to this issue a service descriptor generator component will be added to the WsGenerator tool. In order to preserve the unified interface of WsGenerator the new component will provide solution for all Web Service types. Besides this new component also support for authentication and JSON-RPC [10] will be added in the future.

The ***HugeIndexOfFiles*** web service was created in order to test the ***WSGenerator*** component. In order to create a uniform implementation for the ***HugeIndexOfFiles*** web service the ***WSWrapper*** framework was used. The ***HugeIndexOfFiles*** service can be easily integreated into any application by using the ***WSGenerator*** component.

```
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://wadl.dev.java.net/2009/02 wadl.xsd"
    xmlns:tns="urn:yahoo:yn" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:yn="urn:yahoo:yn" xmlns:ya="urn:yahoo:api" xmlns="http://wadl.dev.java.net/2009/02">
    <grammars>
        <include href="HugeIndexOfFiles.xsd" />
    </grammars>
    <resources base="http:// www.scs.ubbcluj.ro/HugeIndexOfFiles/V1/">
        <resource path="create/{filedb}">
            <method name="POST" id="createid">
                <request>
                    <param name="listDirs" type="xsd:string" style="query"
                        required="true" />
                </request>
                <response status="200">
                    <representation mediaType="application/xml" element="yn:RequestId" />
                </response>
            </method>
        </resource>
        <resource path="update/{filedb}">
            <method name="PUT" id="updateid">
                <request>
                    <param name="listDirs" type="xsd:string" style="query"
                        required="false" />
                </request>
                <response status="200">
                    <representation mediaType="application/xml" element="yn:RequestId" />
                </response>
            </method>
        </resource>
        <resource path="head/{filedb}">
            <method name="GET" id="headid">
                <response status="200">
                    <representation mediaType="application/xml" element="yn:RequestId" />
                </response>
            </method>
        </resource>
        . . . . . .
        <resource path="createResponse/{id}">
            <method name="DELETE" id="createResponseid">
                <response status="200">
                    <representation mediaType="application/xml" element="yn:String" />
                </response>
            </method>
        </resource>
        . . . . . .
    </resources>
</application>
```

FIGURE 6. HugeIndexOfFiles WADL excerpt

REFERENCES

[1] Boian F. M. *Unification of Web Service Technologies,* Proceedings "Zilele Academice Clujene 2010 (ZAC2010)", Ed.Presa Universitara Clujeana,Cluj 2010, ISSN2066-5768, pp. 92-97

[2] Boian F.M. Chinces D, Ciupeiu D, Homorodean D, Jancso B, Ploscar A. *WSWrapper – A Universal Web service generator,* Studia Univ. Babes-Bolyai, Volume LV, Number 4, 2010, pp. 59-69
[3] Boian F.M. *Servicii web; modele, platforme, aplicaii.* Ed. Albastr, Cluj, 2011, pp. 363-369
[4] Boian F.M. *An uniform approach to define and implement the web services; case studies for indexing huge file systems*, Proceedings "Zilele Academice Clujene 2012 (ZAC2012)", pp.1-6
[5] Jancso B. *RESTful Web Services,* Proceedings "Zilele Academice Clujene 2010 (ZAC2010)", Ed.Presa Universitara Clujeana, Cluj 2010, ISSN2066-5768, pp.158-161
[6] Jancso B. *Web Service proxy Generator,* Proceedings "Zilele Academice Clujene 2012 (ZAC2012)", pp.1-6
[7] Marc J. Hadley - Web Application Description Language, 2006
[8] Ploscar A. A *Java Implementation for REST-style web service* ,Proceedings "Zilele Academice Clujene 2010 (ZAC2010)", Ed.Presa Universitara Clujeana, Cluj 2010, ISSN2066-5768,ZAC2010, pp.140-146
[9] Takase T. Makino S. Kawanaka S. Ueno C.F. Ryman A. *Definition Languages for RESTful Web Services: WADL vs. WSDL 2.0.* http://www.ibm.com/developerworks/library/specification/ws-wadlwsdl/index.html
[10] *** JSON-RPC, http://json-rpc.org
[11] *** REST principles, http://www.ics.uci.edu/∼fielding/pubs/dissertation/rest_arch_style.htm
[12] *** SOAP, http://www.w3.org/TR/soap
[13] *** WSDL, http://www.w3.org/TR/wsdl
[14] *** WSDL2.0 HTTP Binding Extension, http://www.w3.org/TR/wsdl20-adjuncts/#http-binding
[15] XML-RPC, http://xmlrpc.scripting.com/spec.html
[16] XRDL, http://code.google.com/p/xrdl

Babeş-Bolyai University, Department of Computer Science, 1 M. Kogălniceanu Street, 400084 Cluj-Napoca, Romania
E-mail address: florin@cs.ubbcluj.ro
E-mail address: bea.jancso@yahoo.com

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVII**, Number 3, 2012

AN EVALUATION OF COLOR SPACES USED IN SKIN COLOR DETECTION

IOAN-GABRIEL MIRCEA

Abstract. The purpose of the present paper is to determine the best choices for color segmentation techniques as basis for skin detection and to review the main approaches in subsequent texture segmentation. The main goal is to determine which color space offers the best clustering for the skin color space and therefore a compared analysis of the most popular color spaces used in the literature is conducted. The results are statistically processed in order to offer the possibility of significant conclusions to be drawn. However, the color segmentation alone does not provide efficiently accurate results when it comes to skin detection so it needs to be followed by efficient texture segmentation that would reduce false positives.

1. INTRODUCTION

An efficient skin detection software is a very useful computer vision tool, either as a stand-alone product or as a starting point for several other more complex software. Such a tool needs to be efficient with respect to both time and resources especially if it is part of a lightweight software intended for the average computer user. Skin detection can be viewed as the core process of more elaborate actions such as face and figure recognition, motion detection, explicit content filtering etc. and its efficiency directly impacts the overall process. This paper is intended to be a sketch for the devise of an efficient skin detection tool that is efficient enough to be easily run on today's usual hardware.

Before the actual development of the skin detection software, many preliminary choices need to be made. After thoroughly browsing the current bibliography on the matter one could conclude that, in fact, skin detection is usually conducted as a sequence of filtering actions upon the pixels of an image: first, only the pixels in the image that have skin-like color are selected,

Received by the editors: September 17, 2012.

2010 *Mathematics Subject Classification.* 68U10.

1998 *CR Categories and Descriptors.* I.4.6 [**Image Processing**]: Segmentation — *Pixel classification*.

Key words and phrases. computer vision, skin detection, color space, texture.

but usually this segmentation gives a high false positive rate and this rate is improved by a subsequent texture filtering stage which selects only the portions of the image which have skin-like texture. If the result obtained after the fore mentioned filters have been applied is still unsatisfactory, additional filters may be employed to reduce the false positive rate - FAR. With this in mind, the current article tries to find which is the best way to conduct color segmentation.

First, a brief overview of different skin color detection techniques is presented. The outcome of each technique is however influenced by the choice of the underlying color space used for color modeling and thus the following section in the article presents the most popular five color spaces used in the literature, highlighting their utility in skin color detection.

This introduces the main objective of the paper: establishing which of the proposed color spaces gives the best results in skin detection. A statistical evaluation is conducted and the results are commented, in order to assess whether statistically significant conclusions may be drawn from the evaluation outcomes.

Finally, as a prelude to an eventual follow-up article certain techniques for texture segmentation encountered in the literature are presented since they would represent the object of the future compared analysis and conclusions are drawn.

2. SKIN COLOR DETECTION

The literature presents several directions for skin color detection which are highlighted thereafter. The first direction consists in explicitly defining the boundaries of the skin color cluster inside the color space. It is a fast method but it requires a good choice for a color space and an efficient set of decision rules that would help achieve high enough detection rates. This approach was used in papers such as [13] and [5] which emphasize how critical the choice for a color model and a set of decision rules is for this kind of approaches.

Another direction is represented by the estimation of the skin color probability distribution based on information obtained from training data. One of the most thorough papers in this respect is [8], referencing a research conducted at The Cambridge Research Laboratory of Compaq Computer Corporation. Having access to a wide database of internet images, a statistical skin color model was created by use of a simple histogram technique. From a total of more than a billion pixels a statistical skin model was constructed by use of two histograms: for skin and non-skin pixels. The likelihood of a pixel being a skin pixel or not was computed via a standard likelihood ratio:

$$\frac{P(rgb|skin)}{P(rgb|\neg skin)} > Ø, 0 < Ø < 1, P(rgb \mid skin) = \frac{s[rgb]}{Ts}, P(rgb \mid \neg skin) = \frac{n[rgb]}{Tn}$$

where s[rgb],n[rgb] represent the skin,respectively non-skin pixels in the histogram bin for a certain color rgb, Ts,Tn are the total number of pixels in the skin ,respectively non-skin histograms and Øis the acceptance threshold.

The likelihood of a pixel being classified either as a skin or non-skin pixel can also be expressed using the Bayes rule, that aids the computation of "a probability of observing skin, given a concrete color value" [16]:

$$P(skin \mid c) = \frac{P(c|skin)P(skin)}{P(c|skin)P(skin)+P(c|\neg skin)P(\neg skin)} \geq Ø$$

These approaches are usually employed when a huge amount of training data can be obtained.

A different way of establishing skin color distribution is by constructing statistical models for skin color likelihood by use of 2-dimensional Gaussian models or by the mixture-of-Gaussians model. This approach is used when the training data is scarcer but still enough to offer viable information to construct the likelihood function. This gives a statistical measure of "how skin-like a certain color is" [10] using the following formula:

$$P(c \mid skin) = \frac{1}{2|\Sigma S|^{1/2}} e^{-\frac{1}{2}(c-\mu_S)^T \Sigma S^{-1}(c-\mu_S)}$$

where

$$\mu_S = \frac{1}{2}\sum_{j=1}^{n} c_j \text{ and } \Sigma S = \frac{1}{n-1}\sum_{j=1}^{n}(c_j-\mu_S)^T(c_j-\mu_S)^T$$

are the mean and co-variance measures obtained from the training data which contains n color samples - c_j

The mixture-of-Gaussians method employs a number of k mixture components for better approximation of the underlying data:

$$P(c \mid skin) = \sum_{i=1}^{k} \pi_i P_i(c \mid skin)$$

The performance boost achieved by using this latter approach in comparison with the single Gaussian approach is perfectly illustrated in [3]. Another important reference on the matter is [6] which uses Expectation Maximization as a technique for parameter adaptation.

According to [16] , the disadvantage with the first approach is that a good set of rules is difficult to find empirically, while the non-parametric approach requires a huge amount of training data which needs to be thoroughly selected and the parametric approach ignores the non-skin statistics from their model which makes them prone to high false positive rates.

3. MOTIVATION FOR COLOR SPACE EVALUATION

Deciding which colorspace gives the best results with respect to skin detection is of utmost importance to the entire skin detection purpose. An efficient chromatic segmentation is not only a good start towards an efficient detection rate but since the color analysis is both the first and the most important part of the detection process and therefore the impact of the choice of color space for representation can deeply influence the overall efficiency of the process. Therefore, a statistical evaluation of the efficiency of the main color spaces used in skin detection is not only justified but also relevant and important to the field. The results obtained from the presented comparison may offer valuable statistically significant information to researchers and developers regarding the viable options for color spaces in the field of skin detection.

It is optimal that the skin cluster formed in the color space is contiguous and somehow compact and although [8] proposed the classic RGB as color space, there are different other color spaces that may express skin clusters in a more compact way, the most popular of which are presented hereafter.

4. COLOR SPACES FOR SKIN DETECTION

After a review of the bibliography, a few color models stood out as being the most popular with respect to skin detection. An important aspect when constructing a color space for skin detection is luminance isolation from the chrominance. Luminance is not considered important when it comes to skin; what we are interested in is chroma since light incidence may alter the luminance of a skin pixel but its chrominance may still lay within skin color boundaries. This aspect is very important when it comes to skin detection since the elimination of the light component would also drastically reduce the size of skin cluster in the color space.

4.1. **Normalized RGB.** The normalized RGB space is computed from the RGB space by obtaining pure colors in two dimensions r, b - from the RGB components:

$$r = \frac{R}{R+G+B} \text{ and } b = \frac{B}{R+G+B}$$

In this way we reduce the color space by expressing the blue and red components as a function of all the RGB components, thus eliminating a dimension in representation. It is useful for a color space to represent the skin space as compact as possible and normalized RGB offers a good clustering. As a result, normalized RGB is successfully used for skin detection - see [2].

4.2. **HSV.** Another popular color space, used in many papers is HSV (Hue-Saturation-Value), a cylindrical color space. Hue defines the dominant color of an area; saturation measures the colorfulness of an area in proportion to

its brightness [14]. The Value represents the luminance of the color and is not of interest to our purpose. This separation of luminance gives a more clear representation of the chrominance in the Hue-Saturation space than in the previous normalized RGB method. HSV can be obtained from RGB by application of the following rules:

$$H = \arccos\left(\frac{\frac{1}{2}((R-G)+(R-B))}{\sqrt{((R-G)^2+(R-B)(G-B)}}\right)\ S = 1 - \left(3\frac{\min(R,G,B)}{(R+G+B)}\right)\ V = \frac{1}{3(R+G+B)}$$

4.3. **YCbCr.** "YCrCb is an encoded nonlinear RGB signal, commonly used by European television studios and for image compression work" [16]. This is a clear choice for skin detection since it separates luminance efficiently and it is easy to be obtained from RGB and vice versa:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

The difference between this color space and HSV is that, due to the easy transformation to and from the RGB space it spreads the values of its Cr and Cb components in a range between 0 and 255 giving a clear choice for the number of bins needed in an eventual histogram approach, while the cylindrical shape of the HSV space requires a linear transformation in order to construct the histogram bins.

4.4. **YCgCr and YCgCb.** Some useful variations of the YCbCr color space are presented in [17]. These two color spaces are similar to YCbCr but substitute R-Y and B-Y respectively with G-Y representing good alternatives for skin segmentation. They can be obtained from RGB values normalized to [0,1] through the following transformations:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 65.481 & 128.553 & 24.966 \\ -81.085 & 112 & -30.915 \\ -37.797 & -74.203 & 112 \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 65.481 & 128.553 & 24.966 \\ -81.085 & 112 & -30.915 \\ 112 & -93.786 & -18.214 \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

These two spaces promise to offer good results in skin detection since they include all the advantages of the YCbCr space enhanced by the additional filtering efficiency of a combined approach.

5. Related Approaches

Although the performance of different color spaces on skin detection has already been assessed in papers such as [1], [4] or [15], the present approach tackles the problem in a slightly different manner since the present goals are different. The research presented in [1] aimed to prove that, if an optimal detector was employed for any of the color spaces - RGB, YCbCr, HSV, CbCr - they would give the same detection rate, but, in contrast, the present approach analyzes only 2-dimensional spaces and evaluates their performance on the same benchmark. Paper [4] follows the same purpose as the present research but it does not include color spaces such as HSV, YCgCr and YCgCb which have obtained good scores in the present evaluation. However the evaluation methodology used in the present paper is, to a certain extent, similar to the one from [4]. A very thorough analysis on the matter is conducted in [15], including a wider rage of color spaces and also considering both the 2-dimensional and 3-dimensional cases, but the paper evaluates the color spaces with respect to several metrics computed on the skin color clusters and not with respect to their impact on the efficiency of the segmentation process, which is the sought purpose of the proposed approach.

6. Color space Evaluation

Given the importance of color space in the process of color skin classification,the purpose of the present paper is to determine which of the previously presented color spaces is the best choice for the construction of skin detection software.

6.1. **Skin Sample Database Construction.** Inspired by the methodology presented in [8], we constructed a database containing only pixels coming from human skin portions selected from various images. Around 1400 skin samples of 50*50 pixels were collected from different images on the internet, depicting skins belonging to different races, from different body areas and in different illumination conditions.

Although a more standardized benchmark might have been used for the research, the choice for a custom benchmark is influenced by the need of a source of skin pixels from various skin-tones and in various illuminations and the pictures provided by the most of the benchmarks -[9]- are usually shot in the same lighting conditions thus reducing the needed generality.

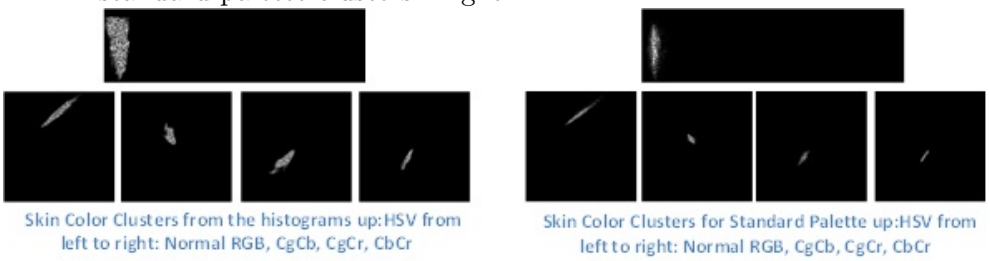
The selected pixel database is used to benchmark the performance of each of the fore mentioned color spaces. In order to do that, histograms are constructed in each color space in order to reveal the skin color clusters. At first the obtained results need to be compared against a chromatic reference for

the human skin. A skin color palette used as reference in the domain of visual arts was chosen - see Figure 1.

FIGURE 1. Skin color palette used as reference.

The shape of the clusters formed in each of the color space depicting all the shades in the reference palette is compared to the corresponding previously constructed histograms.

FIGURE 2. Comparison between histogram clusters - left - and standard palette clusters - right.



The comparison is illustrated by Figure 2, that shows that while the shapes on the left tend to exceed the contour of the ones on the right, they follow the same orientation and the same overall shape, meaning that the constructed color histograms clearly superpose over the colors in the standard palette, giving a reasonably accurate representation of skin color. A total of 3497497 skin pixels were used to construct each histogram.

6.2. **Testing the model.** Once the histograms were constructed and validated against the standard palette, the next step was to test their performance on real images from the internet. A selection of 155 pictures of considerable

size was made such that it contained important amounts of human skin as well as background that might contain skin-like colors. It is important to select a percentage of the pictures to contain materials that resemble skin because we want to test which color space achieves better results even in this situation which influences the false acceptance rate. In order to construct the ideal frame of reference,the human skin pixels in each picture were then manually segmented and a set of another 155 matching pictures containing only those pixels was created. The two paired sets of pictures were used to give the statistical measure of the performance of each color space by the computation of two statistics: the TAR (True Acceptance Rate) and FAR (False Acceptance Rate) of each color space histogram, which are computed by the following rules:

$$TAR = \frac{numberofrealskinpixelsdetectedbythemethod}{numberofpixelsinonly-skinimage}$$

$$FAR = \frac{numberoffalselydetectedpixels}{numberofnon-skinpixelsinonly-skinimage(defaultwhite)}$$

The actual test consisted in computing the True Acceptance Rate and False Acceptance Rate for each color space on all the pictures in the set. A measure of discrepancy between the TAR and FAR was used to assess the performance of each color space on each picture. For each picture, only the best performance was taken into account for measuring which of the color spaces performed best in the most cases. Also, the mean and standard deviation were computed for each color space in order to offer a more compelling picture of their performance. All the significant data obtained from the tests is presented in the table in Figure 3.

FIGURE 3. Table containing test performances for each color space.

average/stdev of TAR	average/stdev of FAR	# of max values
YCbCr		
0.787018561	0.188420976	27
0.2217745	0.171314158	
Normal-RGB		
0.901019324	0.268400145	25
0.158630908	0.199100371	
CgCb		
0.798440968	0.206448848	33
0.215034102	0.17783193	
CgCr		
0.847099832	0.215081078	41
0.193123592	0.188059618	
HSV		
0.846940685	0.277192945	29
0.174494866	0.186165776	

The analysis of the data shows that the two variations of the YCbCr color space, CgCb and CgCr have a slight advantage over the other spaces with

respect to the test score, meaning that these spaces obtained better FARs even on the pictures containing substantial skin-like regions in their background. However, the differences are not significant enough to induce a clear conclusion.

The statistics show that, for the YCbCr space the lowest FAR has been achieved but also the lowest TAR suggesting that this color space is a little bit too drastic with the classification. At the other extreme we have the Normal RGB color space that achieves the highest TAR but also one of the highest FARs meaning it is too permissive. The best scores are achieved by the three spaces that maintain a high TAR and also keep the FAR at an acceptable level regardless of the nature of the picture evaluated.

One important aspect that is considered an advantage on the CgCb , CgCr side is the fact that, as it was mentioned in the description of the HSV color space, its cylindrical shape makes it rather difficult to construct bins for the Hue component. All the conclusions drawn from the analysis of the data in the table are perfectly reflected in Figure 4 which illustrates the performance of color skin classification in all the evaluated color spaces.

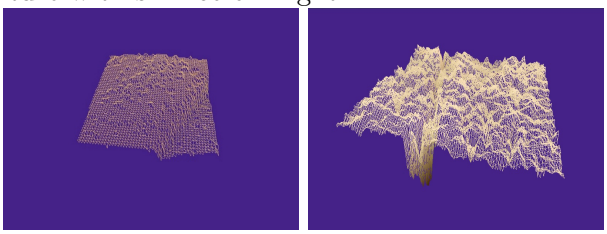
FIGURE 4. Table containing test performances for each color space.

7. Texture Detection - a way of reducing the FAR

As seen in the results of the previous experiment, while the TAR rate was somewhat satisfactory, a FAR of around 20% is achieved in most of the color spaces. This is due to the fact that the usual environment of pictures representing human skin contains materials that have skin-like colors without

being skin. That is why further filtering needs to be applied on the image in order to reduce the False Positives. Many natural materials that resemble human skin in color : sand, some animal's fur, etc. have entirely different textures compared to the texture of the skin. Texture classification can be therefore used to eliminate some of the false positives obtained from color classification. The task is simplified by the fact that usually skin has a smooth texture with slight degradation in luminosity while other materials of skin-like color have a more rugged aspect. A Skin sample is compared to a skin-colored sample with respect to texture in Figure 5.

FIGURE 5. Comparison between skin texture - left - and non-skin texture with skin color -right.



We can accept a slight decrease in the TAR as long as the FAR is reduced significantly after the application of the texture filter, but the choice of the texture measure is crucial. It has to be computationally efficient and also it has to enhance the differences between skin texture and any other texture. The most usual approaches found in current literature are the GLCM, used in [11] or Gabor wavelets [7], but they require a certain amount of computation and parametrization to work efficiently. A non-parametric solution is presented in [12] which uses feature distributions for texture segmentation. The texture classification technique needs to be both exact and fast since the texture segmentation stage is the one the needs to improve the False Acceptance Rate.

8. CONCLUSIONS

After a review of the most popular techniques and color spaces used in skin detection, taking into account the results of the experiment evaluating the performance of different color spaces as basis for skin detection, a choice can be made to use a combined color space CgCr, CgCb in skin color detection since it would bring a good color clustering necessary for imposing a high TAR. However, the obtained FAR is still unsatisfactory if only one form of classification is used and it is of utmost necessity to improve on the FAR by use of an efficient texture filter.

References

[1] Alberto Albiol, Luis Torres, and Edward J. Delp. Optimum color spaces for skin detection, 2001.

[2] Hani K. Almohair, Abd Rahman Ramli, Elsadig A. M, and Shaiful J. Hashim. Skin detection in luminance images using threshold technique skin detection in luminance images using threshold technique abstract. In *International Journal of The Computer, the Internet and Management*, volume 15, pages 25 –32, 2007.

[3] Tiberio S. Caetano, Silvia D. Olabarriaga, and Dante Augusto Couto Barone. Performance evaluation of single and multiple-gaussian models for skin color modeling. In *Proceedings of the 15th Brazilian Symposium on Computer Graphics and Image Processing*, SIBGRAPI '02, pages 275–282, Washington, DC, USA, 2002. IEEE Computer Society.

[4] Jamal Ahmed Dargham Chelsia Amy Doukim and Ali Chekima. Comparison of three colour spaces in skin detection. *Borneo Science*, 24(3):75–81, 2009.

[5] Giovani Gomez and Eduardo F. Morales. Automatic feature construction and a simple rule induction algorithm for skin detection. In *In Proc. of the ICML Workshop on Machine Learning in Computer Vision*, pages 31–38, 2002.

[6] Ming hsuan Yang and Narendra Ahuja. Gaussian mixture model for human skin color and its applications in image and video databases. In *booktitle*, pages 458–466, 1999.

[7] Zhiwei Jiang, Min Yao, and Wei Jiang. Skin detection using color, texture and space information, 2007.

[8] Michael J. Jones and James M. Rehg. Statistical color models with application to skin detection. *Int. J. Comput. Vision*, 46(1):81–96, January 2002.

[9] A. M. Martinez and R. Benavente. The AR Face Database. Technical report, CVC, June 1998.

[10] Bernd Menser and Mathias Wien. Segmentation and tracking of facial regions in color image sequences. In King N. Ngan, Thomas Sikora, and Ming-Ting Sun, editors, *VCIP*, volume 4067 of *Proceedings of SPIE*, pages 731–741. SPIE, 2000.

[11] Anal Kumar Mittra. Automated detection of skin disease using texture features. In *International Journal of Engineering Science and Technology*, pages 4801–4808, 2011.

[12] Timo Ojala and Matti Pietikäinen. Unsupervised texture segmentation using feature distributions. *Pattern Recognition*, 32(3):477–486, 1999.

[13] Peter Peer, Jure Kovac, and Franc Solina. Human skin colour clustering for face detection, 2003.

[14] Charles A. Poynton. *Frequently asked questions about colors*. publisher, 1997.

[15] Min C. Shin, Kyong I. Chang, and Leonid V. Tsap. Does colorspace transformation make any difference on skin detection? In *In IEEE Workshop on Applications of Computer Vision*, pages 275–279.

[16] Vladimir Vezhnevets, Vassili Sazonov, and Alla Andreeva. A survey on pixel-based skin color detection techniques. In *IN PROC. GRAPHICON-2003*, pages 85–92, 2003.

[17] Shi Yuexiang. Zhang Zhengzhen. Skin color detecting unite ycbcg color space with ycgcr color space. note, 2009.

Babeş-Bolyai University, Department of Computer Science, 1 M. Kogălniceanu St., 400084 Cluj-Napoca, Romania

E-mail address: mims0509@scs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVII**, Number 3, 2012

NEW MAJORITY RULE FOR NETWORK BASED CELLULAR AUTOMATA

ANCA ANDREICA AND CAMELIA CHIRA

Abstract. Cellular Automata represent useful and important tools in the study of complex systems and interactions. This study focuses on the evolution of both network topologies and majority rules for the density classification task. Results show that a variation in the classical definition of the majority rule could induce higher performance of the cellular automata.

1. Introduction

Cellular Automata (CAs) are decentralized structures of simple and locally interacting elements that evolve following a set of rules [8], usually based on a regular lattice topology. Watts and Strogatz [9, 10] studied the CAs computation on small-world networks. In [1, 2, 7], small-world type network topologies are evolved starting from an initial population of regular and random structures. In these studies, it is shown that the evolved topologies have better performance for the CA majority problem than regular lattice structures.

In [3] we investigate the evolution and dynamics of small-world networks for CA computation. The density classification task is addressed using network topologies evolved based on a simple standard genetic algorithm. Computational experiments and results indicate that the obtained networks have a competitive performance for CA density classification even when simply the majority rule is applied.

In this paper we extend our research on network based cellular automata by searching for a new majority threshold for the density classification problem. A standard evolutionary algorithm is used in order to evolve both network topologies and majority rules that induce higher CAs performances.

Received by the editors: September 24, 2012.

2010 *Mathematics Subject Classification.* 68-02.

1998 *CR Categories and Descriptors.* I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search – *Heuristic methods*.

Key words and phrases. evolutionary algorithms, density classification task, cellular automata.

2. Majority rule for network based Cellular Automata

The one-dimensional binary-state CA capable of performing computational tasks has been extensively studied in the literature [6, 4, 5]. A one-dimensional lattice of N two-state cells is usually used for representing the CA. The state of each cell changes according to a function depending on the current states in the neighborhood. The neighborhood of a cell is given by the cell itself and its r neighbors on both sides of the cell, where r represents the radius of the CA. The initial configuration of cell states (0s and 1s) for the lattice evolves in discrete time steps updating cells simultaneously according to the CA rule.

One of the most widely studied CA problems is the density classification task (DCT). The aim of DCT is to find a binary one-dimensional CA able to classify the density of 1s (denoted by ρ_0) in the initial configuration. If $\rho_0 > 0.5$ (1 is dominant in the initial configuration) then the CA must reach a fixed point configuration of 1s otherwise it must reach a fixed-point configuration of 0s within a certain number of time steps.

The performance of a rule is given by the classification accuracy of a CA based on the fraction of correct classifications over 10^4 initial configurations selected from an unbiased distribution.

The rule used for network based DCT states that at each time step, each node takes the state of the majority of its neighbor nodes in the graph (if the number of state 1s equals the number of state 0s in the neighbors list then the node is randomly assigned a state with equal probability between 0 and 1).

Therefore, in the classical approach there is a threshold of 0.5 which represents the density of neighbors with the same value that pass their state to the current cell. Let us call it the *majority threshold*. The aim of this paper is to identify a different *majority threshold* that induces higher performance CAs. For example, for a majority threshold of 0.7, it would mean that a cell should receive the state 1 only if there are more than 70% neighbors having the value 1. Our intention is to find whether there is another equilibrium point of the system, different than 0.5.

A basic coevolution scheme is used to evolve both better majority thresholds and network topologies with better performance for CAs. The goal of evolving both types of solutions is to obtain a network more suitable to the new majority rule and vice versa. The performance and the robustness of the obtained network will also be studied.

3. Experimental results

A standard evolutionary algorithm is engaged to evolve both network topologies for cellular automata and new majority thresholds, as described in

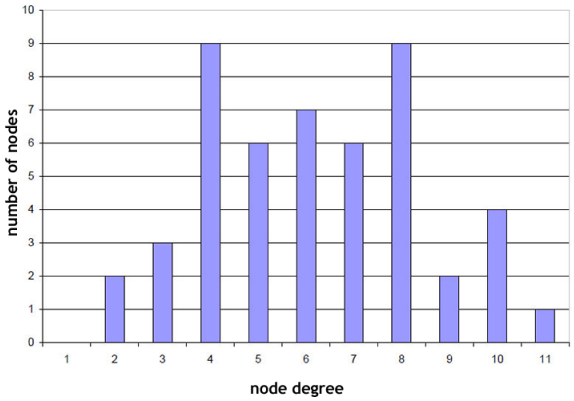


FIGURE 1. Node degree distribution for obtained network topology.

the previous section. There are two populations of individuals that evolve in parallel. There is one population of networks, where an individual is encoded as an array of integers representing nodes and a list of links for each node. The other population consists of individuals representing majority thresholds, therefore encoded as real number between 0 and 1. Both populations have 100 individuals and the number of generations is 100. Standard proportional selection, 10% elitism and weak mutation are applied for both populations. The fitness function is computed as the fraction of correctly classified 100 randomly generated initial configurations.

The evolution process starts with a regular lattice ring where each cell has 4 neighbors. At each generation, the best 20% individuals are fed into the other current population, which takes over the evolutionary process by searching for a better network and a better majority threshold, respectively.

As preliminary experiments, the described algorithm has been applied for cellular automata with 49 cells. In [3] we computed the performance of the obtained network with the fixed majority threshold of 0.5. For 49 cells CAs, the obtained performance was around 0.85. When evolving both network and majority threshold, we have obtained a slightly better performance (0.89) and a majority threshold around 0.6, out of 10 runs of the algorithm. These results indicate the fact that a better performance could be obtained when the majority is more strongly marked.

When analyzing the obtained network one can see that the average node degree is 6, which resembles the neighborhood of 7 usually used for regular lattice CAs. Figure 1 presents the distribution of node degrees in the obtained network. It can be noticed that the node degrees are close to the average.

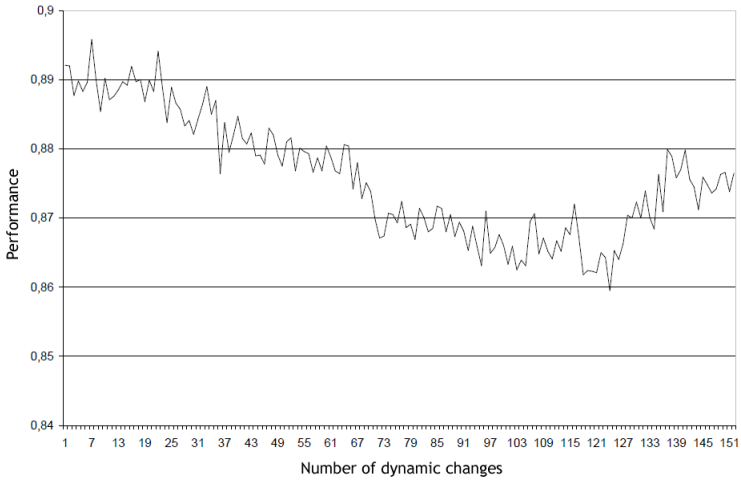


FIGURE 2. The performance of the network that is subject to dynamic changes with probability $p = 0.5$.

In order to evaluate the robustness of the evolved topology, the network has been subject to dynamic changes understood as random removal or addition of network links. The number of dynamic changes equals the number of edges in the network. At each step, a randomly selected link is either removed or added to the network with probability p and the resulting network is again evaluated with respect to the performance. Figure 2 depicts the performance of the considered network subject to dynamic changes with probability $p = 0.5$. The network is still able to trigger good performances on the DCT i.e. above 0.85. The high fluctuation in performance observed in the obtained network might be due to the fact that the algorithm has been able to find a good solution for this small network in its 100 generations, and a certain degree of destabilization is induced when the network is subject to further dynamic changes.

We further analyze the performance of the evolved network when the probability p of dynamic change ranges from 0.1 to 1 (see Figure 3). One can see that the performance for DCT remains good even when changes are induced with high probability. This is an indication of the robustness of the evolved network topology and its capability to obtain better results for the DCT (which can be potentially further improved in connection with more sophisticated rules compared to the fixed majority rule).

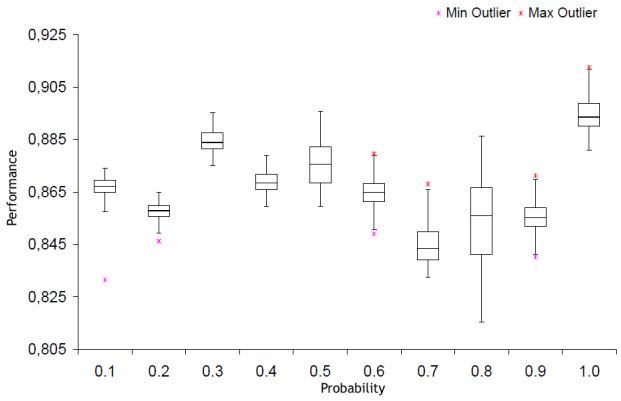


FIGURE 3. The performance of the 49 cells network subject to dynamic changes with different probabilities.

4. Conclusions and future work

A new approach to the standard understanding of majority rule for network based cellular automata has been proposed in this paper. On this purpose, a basic coevolution scheme has been applied in order to evolve a higher performance majority rule for network based CAs, and also a more suitable network topology. Preliminary results of our study led to the conclusion that there is a majority threshold different than 0.5 which might induce a higher performance for network based CAs. The performance of the obtained network topology is analyzed when the network is subject to dynamic changes. Our study will be further improved by considering CAs having more cells, especially the case of 149 cells which is considered by most of the studies found in the literature. We also plan to investigate the role of second degree neighbors when applying the majority rule for network based CAs, in order to further improve the CAs performance.

5. Acknowledgments

This research is supported by Grant PN II TE 320, Emergence, auto-organization and evolution: New computational models in the study of complex systems, funded by CNCSIS, Romania.

References

[1] Darabos, C., Giacobini, M., Tomassini, M., Performance and Robustness of Cellular Automata Computation on Irregular Networks. Advances in Complex Systems 10: 85-110 (2007)

[2] Darabos, C., Tomassini, M., Di Cunto, F., Provero, P., Moore, J.H., Giacobini, M., Toward robust network based complex systems: from evolutionary cellular automata to biological models, Intelligenza Artificiale 5(1): 37-47 (2011)
[3] Gog, A., Chira, C., Dynamics of Networks Evolved for Cellular Automata Computation, HAIS (2), Lecture Notes in Computer Science, Springer, Vol. 7209, pp. 359-368 (2012)
[4] Mitchell, M., Thomure, M. D., Williams, N. L.: The role of space in the Success of Co-evolutionary Learning. Proceedings of ALIFE X - The Tenth International Conference on the Simulation and Synthesis of Living Systems (2006)
[5] Oliveira, G.M.B., Martins, L.G.A., de Carvalho, L.B., Fynn, E., Some investigations about synchronization and density classification tasks in one-dimensional and two-dimensional cellular automata rule spaces, Electron. Notes Theor. Comput. Sci., 252 (2009), pp. 121-142.
[6] Tomassini, M., Venzi, M.: Evolution of Asynchronous Cellular Automata for the Density Task. Parallel Problem Solving from Nature - PPSN VII, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2439, 934-943 (2002)
[7] Tomassini, M., Giacobini, M., Darabos, C., Evolution and dynamics of small-world cellular automata, Complex Systems 15 (2005), 261284.
[8] Wolfram, S., Theory and Applications of Cellular Automata, Advanced series on complex systems, World Scientific Publishing, 9128 (1986).
[9] Watts, D.J., Strogatz, S.H., Collective dynamics of 'smallworld' networks, Nature 393 (1998), 440442.
[10] Watts, D.J., Small Worlds: The Dynamics of Networks Between Order and Randomness, Princeton University Press, Princeton, NJ (1999)

Department of Computer Science, Babes-Bolyai University, Kogalniceanu 1, 400084 Cluj-Napoca, Romania
E-mail address: {anca,cchira}@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVII**, Number 3, 2012

ANALYZING AND TUNING USER QUERIES TO SEARCH ENGINES

DARIUS BUFNEA

Abstract. There are certain situations when a web site's visitor using a search engine as a referrer will not be correctly redirected to the desired product or information page, although such a page exists within the web site. This paper presents a solution for a web site to locally further analyze and tune user queries to search engines in order to lead the user to the page describing the product he is interested in. This can improve the visibility of products within the website and in the same time increase its conversion rate. Misdirecting the user and failing in satisfying his interest will be reflected in the revenue amount of a website; on the contrary, satisfied visitors can become potential clients and on a long time scale can improve the website's ranking within a search engine.

1. Introduction

As web technologies developed and their usage on business oriented application as e-commerce have increased, the research community and in the same time private enterprise studies have led to research for solutions to maximize the impact of the web interaction with an online consumer. The impact of the web interaction with an online consumer can be quantified by using certain variables such as: time spent by the visitor within the website, number of products description visualized, conversion rate, number of ordered products or number of displayed commercial ads.

In this paper we present a solution for analyzing and tuning user queries to search engines in logic implemented at web application's tier.

Web sites and web applications in Internet depend on a large scale on visitors sent by search engines. For a regular web site, the percent of these visitors can be as high as 70 to 80 percent from the total number of visitors. Our presented method enhances a search engine's effort to redirect the

Received by the editors: September 17, 2012.

2010 *Mathematics Subject Classification.* 68U35, 68M11.

1998 *CR Categories and Descriptors.* H.3.5 [**Information Storage and Retrieval**]: Online Information Services – *Web-based services*.

Key words and phrases. Web Referrer, Search Query, User Experience.

visitor to the desired page, i.e. the page that contains information about a product searched by the visitor. This method should not be confounded with Search Engine Optimization (SEO) techniques. Our method rather complements these approaches. There are certain situations when a search engine might fail in redirecting the user to the correct web page, even if such a page exists within the website. A missed search redirected user may leave the website and give up in further searching for the product, even if the web site is offering a search feature. The short term impact is the loss of a possible customer. On a long time run, such a visitor repeated behavior may lead to web site's position demotion in a search engine's results for certain keywords. It is well known that Google for example permanently adjusts their order of return search results based on previous users' experience and feedback.

2. Previous Work

From the mid 90s as web usage increased, numerous different techniques were proposed in order to maximize the interaction between a web site and an online user visiting the site. These techniques are based on different approaches such as data mining, pattern discovery or artificial intelligence. All these techniques aim the same goal: getting the user more closely to the product or information he is interested in. Some studies analyze market basket data ([1], [2]), an approach useful for determining and generating relevant web content for similar consumer-desired products based on previous collected data. Another direction of research ([3], [4]) is pattern discovery across a user's visit path within a website, approach that can be used to create dynamically adaptive web sites based on users' online behavior patterns.

In order to maximize the interaction with its visitors, a website with a relative constant audience can rely on user based personalization techniques via cookies or server side saved configuration settings.

More recently, semantic web approaches, although designed to increase interaction between websites and web applications, can also improve a product visibility within a website for a human person trying to reach that product's specification page. For example, GoodRelations [5] might help a referrer in locating pages of other web sites describing similar products.

3. Contributions

In this paper we focus on those situations when the visitor is sent to the web site from a search engine, i.e. the search engine is a referrer for the web site. There are situations when a search engine will fail to redirect visitors to the most search query related page within the web site. A typical scenario is the visitor landing on the root page of the web site, simply because news about

a product he's interested in is posted on that page. It's very common for web site's root page to have a higher page rank than an in-site page containing a product's description.

Our idea is based on the assumption that business logic running at the web server can extract from the user's query more semantic information than the search engine does. This semantic information is used to redirect user to a more search query related page than the one user has landed on. This approach is similar in many aspects with Search Engine Optimization (SEO) techniques, but while SEO is used to improve a web site visibility in search engines, our method is used to improve a product visibility within the website itself.

3.1. **Technical Fundamentals.** A significant number of requests for resources sent to a web server are accompanied by a `referer`[1] HTTP header defined in RFC 2616. In fact, all requests, not directly typed in a web browser by the user, are accompanied by such a header. This header is added by the web browser to requests made by following links from one page to another, even if pages are hosted on different domains. The header is also present in requests for resources hosted on the same domain as the referrer, for e.g. in request for images needed for correctly rendering a web page.

The `referer` HTTP header is also present in requests for web pages made via a search engine. Although, search engines typically are not hot linking to other websites - rather they redirect the visitor using HTTP redirects - the URL of the search page within the search engine is preserved as a referrer in the request made for the visited landing page (i.e. the web pages where the user is redirected by the search engine). The referrer URL within the search engine always includes the user's search query as a value to a specific search engine dependent attribute. For example, a typical `referer` header for a request made via Google will include a `q` attribute whose value is set to the user's query.

With the knowledge about the user's query, we can locally make better assumption about its semantic than a search engine does. The query might be used to identify if the search engine correctly redirects the user to the most query related page and, if this is not the case, locally redirect him to a more content related page to his query.

3.2. **The search query module.** For a better understanding of our approach we'll give some real world examples about the inappropriate behavior of a search engine in our context.

[1]The name of the header, i.e. `referer` is deliberately misspelled in the RFC 2616 because of historical considerations. Correct spelling should be referrer.

a) A news site containing on its main page the latest news is indexed by a search engine. Subsequently in time, a visitor is landing on the news site's main page from the search engine, his query being related to news posted on the main page at crawling time. If the news that might catch the visitor's attention is at visiting time demoted from the main page, the visitor will not find it and might bounce and go on to the next site within the search result.

b) Someone is looking for flights from London to New York using a search engine. The search engine can easily redirect a user to a page on the flight operator's web site containing the schedule of all departures from New York to London. Even if the flight operator's site is offering a very accessible search form or a "return flights" button, some users will abandon their search experience (i.e. visit) to the flight operator's web site, simply because they don't see a departure hour from London to New York. Instead, such a user will prefer to jump to the next site in the search result, often a competitor's web site.

c) Some person is looking for a notebook with a built in 3G wireless modem. Once again, a search engine can easily redirect the user to an online store's web site offering for sale a notebook with wireless 801.11 connectivity (i.e. Wi-Fi), a classical dial up PSTN modem and 3 GB of RAM. The users will immediately catch the missed landing page, and might prefer visiting the next site within the search result, even if the above online store is really offering a product the user is interested in.

In all these scenarios a web module run by a web site can make new assumptions about the user's query semantic, further refining the search query.

The main task of this module should be:

- Identify from the referrer request header (in case there is one) if the visit is coming from a search engine, and if this is the case what is the user's query value;
- Parse the user's query and match it to products stored in the database backend server. The matching process should take into consideration different product characteristics such as product name, manufacturer, description, specifications and so on. In the matching process different similarity matching algorithms should be tested. Although we implement such an algorithm in our test module (discussed in different writing up paper) future research is to be done. This algorithm is based on the vector space model and is used to map user's queries to product's attributes within the backend database server. Speed and accuracy of the matching process are critical for our technique's success.
- Redirect the user to the match product's description page through a transparent mechanism. Such a redirection can be implemented using HTTP 3xx headers or by using a client side script to perform the jump

from the missed landing page to the desire one. Preliminary researches we have performed are in favor of the client side scripting approach, mainly in order to avoid bounce rate increasing.

The search query module should be run for every page request coming from a search engine result index. Some solutions for running this module are either call it from within a master page or implement it as a server side technology independent filter that intercept the user's request before it reaches the final resource (i.e. the missed landing page).

For common, very often used queries, this module can also cache the match product in order to increase and improve the web page serving speed. Other important aspects that should be taken into consideration by such a module are: words order in user's query, frequently typos or words abbreviation often used by visitors in their queries.

3.3. **Experiment.** In order to assert the method presented in this paper, we made an experiment over a commercial in production web server. This enterprise web server was also used for implementing and testing our method. Because of confidential agreements we cannot disclosure the name of the web server, but we have the approval for using any obtain results for research purpose in scientific papers.

The obtained results support the idea presented in this paper, but actual values are extremely site depended. They were obtained for a very search engine friendly web site (i.e. a web site that presents its content semantic in an easy understandable way by the search engine). Actual percent of these values might be different from site to site being affected by some parameters such as: site niche, the degree of site search engine optimization and advertising campaigns. The experiment was conducted over one month by a web application server module, but the measurements were also confirmed using standard tools such as Google Analytics [7].

In this experiment we were interested in:

- Percent of visitors that have landed on our web site via a search engine using a query string that can be matched to a product. For previous given examples, a product can be a notebook, a flight from London to New York or a news title. The matching process is important in order to further determine if the user's landing page is a good one, i.e. it describes the product the user is interested in, or not.
- Percent of web visitors that have landed via a search engine on a wrong page (i.e. page not describing the product they are interested in);
- Percent of visitors that have landed on a wrong page and they subsequently used website's available search form or navigate further to find information about their product of interest;

– Percent of visitors that have landed on a wrong page and they subsequently abandoned their visit on the web site (i.e. they bounce).

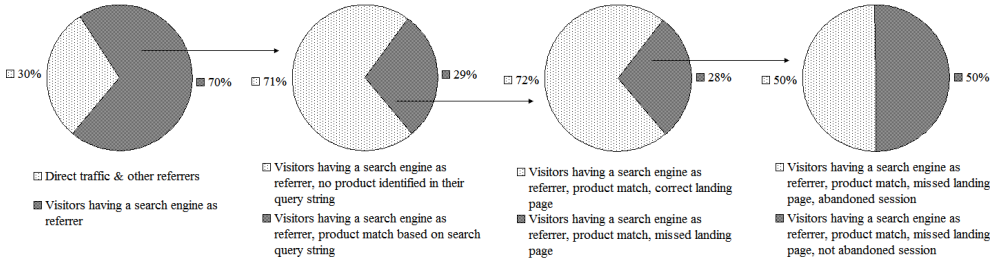


FIGURE 1. Visits, product match and users' behavior for a test web site

We are especially interested in the percent of visitors that have landed on a wrong page and they subsequently abandoned their visit on the web site, those visitors becoming potentially lost customers. Our method increases product match ratio based on user's query and subsequently increases conversion rate by correctly redirecting user's browser to the desired page.

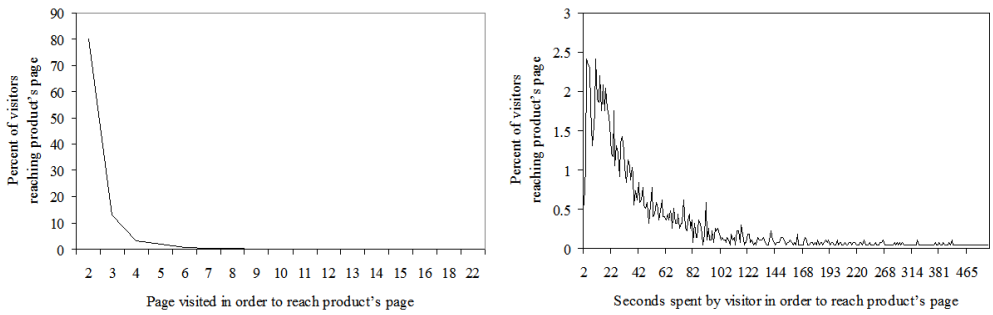


FIGURE 2. Page visited in order to reach products page (left) Seconds spent by visitor in order to reach products page (right)

Furthermore, we are also targeting visitors that have landed on a wrong page and subsequently they are continuing their session looking for information about a specific product. Figure 2 depicts this behavior, showing the number of visited pages and the time spent in searching for a specific product by percent of visitors - percent from the total amount of missed landing users

that are not abandoning their session. Our method helps in reducing user's effort in reaching product information by decreasing the total number of pages that he has to visits or the amount of time he spent searching that information.

	Module off	Module on
Bounce rate	44.74%	41.75%
Visits coming from search engines	70.22%	70.47%
Visitis having a search engine as referrer and a product was identified in user's query string	20.37%	20.43%
Visits having correct landing pages	14.66%	20.43%
Visits having missed landing pages	5.71%	-
Visitors that have landed on a wrong page and they subsequently continued visiting our website	2.86%	-
Bounce visitors because of missed landing pages	2.85%	-
Conversion rate	2.23%	2.64%

FIGURE 3. Web site behavior with our module set to off / on

Figure 3 presents the results obtained with our module set to off for one month and to on for another month. When set to off, in fact it was running in a special state performing no redirects, but only measurements and collecting the above statistical data. When set to on, in case of an initially missed landing page, it consequently redirects the user to the page describing the product identified in user's search query. A major benefit of running the proposed module is reducing of the total bounce rate. This is done by eliminating visitors that bounce because they land on wrong pages. In fact, even if in a missed landing pages situation, each session will count at least two requests, one for the initially missed lading page and one for the page describing the product visitor is interested in. By leading visitors more precisely by their interest, an increase in the conversion rate can also be observed, the potentially lost customers being transformed in potentially gain customers.

Note: All percent values in figure 3 are relative to the total amount of site traffic. We counted only visitors using key words that we successfully match against a product from the web site's offer. We were not interested on visitors using generic key words such as company's name or web site's name as the search query.

4. Conclusions and future work

In this paper we have advanced a method for analyzing and tuning user queries to search engines. Our method helps visitors in reaching the product

information they are interested in and in the same time it makes a product more visible within the website. We are currently focusing our efforts in evaluating different similarity algorithms based on the vector space model in order to improve the matching process between search queries and products description. The speed of the matching process is extremely important, because any delay in serving the desired page to a client may be reflected in the Page Rank of a website, site speed being one of the criteria that Google uses in computing a website Page Rank.

References

[1] Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman, Shalom Tsur, *Dynamic itemset counting and implication rules for market basket data*, in Proceedings of the 1997 ACM SIGMOD international conference on Management of data 1997 (SIGMOD '97), New York, NY, USA, pp. 255-264.

[2] Alexandros Nanopoulos, Yannis Manolopoulos, *Efficient similarity search for market basket data*, The VLDB Journal 11, 2 (October 2002), pp. 138-152.

[3] Alina Campan, Darius Bufnea, *Automatic Support for Improving Interaction with a Web Site*, in Studia Universitatis Babeş-Bolyai, Informatica, Vol. XLV(2), pp. 95-103, 2000.

[4] Poonam Goyal, Navneet Goyal, Ashish Gupta, T. S. Rahul, *Designing self-adaptive websites using online hotlink assignment algorithm*, in Proceedings of the 7th International Conference on Advances in Mobile Computing and Multimedia 2009 (MoMM '09), ACM, New York, NY, USA, pp. 579-583.

[5] Martin Hepp, *GoodRelations: An Ontology for Describing Products and Services Offers on the Web*, in Proceedings of the 16th International Conference on Knowledge Engineering and Knowledge Management (EKAW2008), Acitrezza, Italy, September 29 - October 3, 2008, Springer LNCS, Vol. 5268, pp. 332-347.

[6] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, *Hypertext Transfer Protocol – HTTP/1.1*, RFC 2616, June 1999.

[7] *Google Analytics*, Google Inc., <http://www.google.com/analytics/>.

Babeş-Bolyai University, Department of Computer Science, 1 M. Kogălniceanu St., 400084 Cluj-Napoca, Romania

E-mail address: bufny@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVII**, Number 3, 2012

XML SCHEMA REFINEMENT THROUGH FORMAL CONCEPT ANALYSIS

KATALIN TUNDE JANOSI-RANCZ AND VIORICA VARGA

Abstract. As XML becomes a popular data representation and exchange format over the web, XML schema design has become an important research area. Formal Concept Analysis (FCA) has been widely applied in many fields recently. In this paper, we propose the application of FCA to find functional dependencies (FDs) in XML databases. Our work is based on the definitions of the Generalized Tree Tuple, XML functional dependency and XML key notion presented by [22]. We propose a framework which parses the XML document and constructs the Formal Context corresponding to the flat representation of the XML data. The obtained Conceptual Lattice is a useful graphical representation of the analyzed XML document's elements and their hierarchy. The software also finds the keys and functional dependencies in XML data, which are attribute implications in the constructed Formal Context. The scheme of the XML document is transformed in GTT-XNF using the detected functional dependencies.

Keywords and phrases: XML design, Formal Concept Analysis, XML Functional Dependency, XML Normal Forms.

1. Introduction

In the last few years several papers discussed the relationship between Formal Concept Analysis (FCA) and relational databases [8, 10]. Our approach intends to extend these results by reformulating the XML functional dependency inference with an FCA viewpoint.

FCA is a mathematical theory of concept hierarchies which is based on Lattice Theory. It is used as a technique for data analysis, knowledge representation; it is a useful tool to represent knowledge contained in a database.

Designing XML data means to choose an appropriate XML schema, which usually come in the form of DTD (Document Type Definition) or XML Scheme. A big number of classical database subjects have been reexamined in the XML

Received by the editors: October 8, 2012.

2010 *Mathematics Subject Classification.* 68P15, 03G10.

1998 *CR Categories and Descriptors.* H.2.1 [**Database Management**]: Logical design — *Normal forms*.

context [7, 1, 18, 17, 3] because XML became more and more popular. Discovering XML data redundancies from the data itself becomes necessary and it is an integral part of the schema refinement (or re-design) process.

Recently, there were several attempts to define XFDs (see [6, 16, 9, 19]) but in general, these approaches have different semantics regarding the *tree tuple* and *closest node* XFDs and do not preserve the semantics of FDs when relational data is mapped to XML via arbitrary nesting.

Functional dependencies (FDs) are a key factor in XML design. Our paper proposes a framework to mine FDs from an XML database; it is based on the notions of Generalized Tree Tuple, XML functional dependency and XML key notion presented by [22]. Our contribution is the construction of the formal context for a tuple class or the whole XML document. Non-leaf and leaf level elements (or attributes) and corresponding values are inserted in the formal context, then the concept lattice of the XML data is constructed too. The obtained Conceptual Lattice is a useful graphical representation of the analyzed XML document's elements and their hierarchy. The software also finds the keys in the XML document. The set of implications resulted from this concept lattice will be equivalent to the set of functional dependencies that hold in that database.

This paper is an extended version of the short conference paper that appears as [12]. The key additions to this journal paper are Section 2, which discusses related work, Section 3 presents the necessary definitions of Functional dependency for XML data and GTT-XNF normal form. In Section 4 it is explained in detail how these two techniques are combined to mine functional dependencies. As a novelty of this article is the transformation of a given XML Scheme to a scheme in GTT-XNF form. Finally, Section 5 summarizes the conclusions and future work.

2. Related work

The first authors who presented the problem of finding functional dependencies in many-valued context were Ganter and Wille [8]. Different authors [14, 15] use FCA concepts and methods like agree sets, maximal sets and closed sets, which are closely related to the concept of closed sets and generators, described previously, to avoiding the transformation of the original database. The authors use these concepts to find efficient algorithms to extract functional dependencies from a relational database. [4] studied the lattice characterization and its properties for Armstrong and symmetric dependencies. Hereth has already described the relationship between FCA and functional dependencies in [10], he has introduced the formal context of functional dependencies. In this context, implications hold for functional dependencies. The paper [11]

presents an FCA based approach to detect functional dependencies in a relational database table.

The first authors who formally defined XML FD and normal form (XNF) were Arenas and Libkin introducing the so-called *tree tuple* approach [1]. In [13] and [18], the authors used a *path-based* approach and built their XML FD notion in a way similar to the XML Key notion proposed in [5].

Yu and Jagadish [22] show, that these XML FD notions are insufficient and propose a Generalized Tree Tuple (GTT) based XML functional dependency and key notion, which include particular redundancies involving set elements. Based on these concepts, the GTT-XNF normal form is presented too.

In later work [2], Arenas and Libkin provided a formal justification for the use of XNF in XML database design, using the classical information theory approach. A measure of the information content of data (independent of updates and queries) is introduced, as entropy of a suitably chosen probability distribution. A formal definition of a well designed XML schema was given and the fact that XNF is both a necessary and sufficient condition for an XML schema to be well designed was proved.

Vincent et al. in [17] investigated the problem of justifying XML normal forms [18], in the terms of *closest node* XFDs using redundancy elimination. In [17], a normal form for XML documents is proposed and it has been proved to be a necessary and sufficient condition for the elimination of redundancy.

3. Functional dependency for XML data

Arenas and Libkin introduced first the so-called *tree tuple* notion in [1], they have defined functional dependency in XML data and XNF normal form for XML. Redundancies in XML data have several distinct features due to the heterogeneous nature of XML data, which makes them richer in semantics as compared with redundancies in relational data. Yu and Jagadish [22] show, that the XML FD notion introduced by [1] doesn't include all possible features of an XML document and propose a Generalized Tree Tuple (GTT) based XML functional dependency and XML key notion, which include particular redundancies involving set elements. They propose the GTT-XNF normal form too based on these notions. As [22] we treat leaf level elements and attributes in the same manner. The definitions of this section are based on [22].

Definition 1. (Schema) *A schema is defined as a set* $S = (E, T, r)$*, where:*

- E *is a finite set of element labels;*
- T *is a finite set of element types, and each* $e \in E$ *is associated with a* $\tau \in T$*, written as* $(e : \tau)$*,* τ *has the next form:*
  $\tau ::= \texttt{str} \mid \texttt{int} \mid \texttt{float} \mid \texttt{SetOf}\ \tau \mid \texttt{Rcd}[e_1 : \tau_1, \ldots, e_n : \tau_n]$*;*

- $r \in E$ *is the label of the root element, whose associated element type can not be* `SetOf` τ.

This definition contains some basic constructs in XML Scheme [20]. There are a lot of other datatypes defined at [20], but it is not relevent for our problem to enumarate them. Types `str,int` and `float` are the system defined *simple types* and `Rcd` indicate *complex scheme elements* (elements with children elements). Keyword `SetOf` is used to indicate *set schema elements* (elements that can have multiple matching data elements sharing the same parent in the data). We will treat attributes and elements in the same way, with a reserved "@" symbol before attributes.

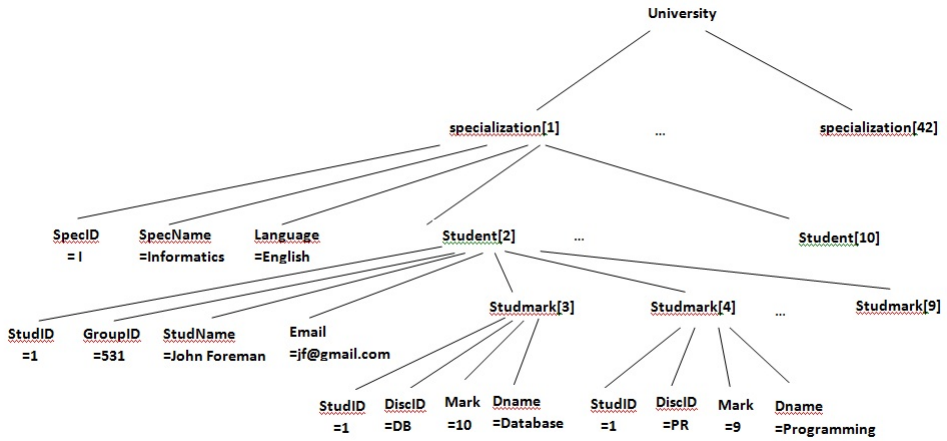


FIGURE 1. Example tree

The examples of this paper are based on XML tree of Figure 1.

Example 1. The scheme $S_{University}$ of XML document from Figure 1 is:

```
University:Rcd
   specialization:SetOf Rcd
      SpecID: str
      SpecName: str
      Language: str
      Student: SetOf Rcd
         StudID: int
         GroupID: str
         StudName: str
         Email:str
```

```
Studmark: SetOf Rcd
    StudID: int
    DiscID: str
    Mark: int
    DName: str
```

A schema element e_k can be identified through a path expression, $path(e_k) = /e_1/e_2/.../e_k$, where $e_1 = r$, and e_i is associated with type $\tau_i ::= \texttt{Rcd}\ [..., e_{i+1} : \tau_{i+1}, ...]$ for all $i \in [1, k-1]$. A path is *repeatable*, if e_k is a set element. We adopt XPath steps "." (self) and ".." (parent) to form a relative path given an anchor path.

Definition 2. (Data tree) *An XML database is defined to be a rooted labeled tree* $T = \langle N, \mathcal{P}, \mathcal{V}, n_r \rangle$*, where:*

- N *is a set of labeled data nodes, each* $n \in N$ *has a label* e *and a node key that uniquely identifies it in* T*;*
- $n_r \in N$ *is the root node;*
- $\mathcal{P}$ *is a set of parent-child edges, there is exactly one* $p = (n', n)$ *in* $\mathcal{P}$ *for each* $n \in N$ *(except* n_r*), where* $n' \in N, n \neq n', n'$ *is called the parent node,* n *is called the child node;*
- $\mathcal{V}$ *is a set of value assignments, there is exactly one* $v = (n, s)$ *in* $\mathcal{V}$ *for each leaf node* $n \in N$*, where* s *is a value of simple type.*

We assign a node key, referred to as @key, to each data node in the data tree in a pre-order traversal. A data element n_k is a descendant of another data element n_1 if there exists a series of data elements n_i, such that $(n_i, n_{i+1}) \in \mathcal{P}$ for all $i \in [1, k-1]$. Data element n_k can be addressed using a path expression, $path(n_k) = /e_1/\ldots/e_k$, where e_i is the label of n_i for each $i \in [1, k]$, $n_1 = n_r$, and $(n_i, n_{i+1}) \in \mathcal{P}$ for all $i \in [1, k-1]$.

A data element n_k is called *repeatable* if e_k corresponds to a set element in the schema. Element n_k is called a *direct descendant* of element n_a, if n_k is a descendant of n_a, $path(n_k) = \ldots/e_a/e_1/\ldots/e_{k-1}/e_k$, and e_i is not a set element for any $i \in [1, k-1]$.

In considering data redundancy, it is important to determine the equality between the "values" associated with two data elements, instead of comparing their "identities" which is represented by @key. So, we have:

Definition 3. (Element-value equality) *Two data elements* n_1 *of* $T_1 = \langle N_1, \mathcal{P}_1, \mathcal{V}_1, n_{r1} \rangle$ *and* n_2 *of* $T_2 = \langle N_2, \mathcal{P}_2, \mathcal{V}_2, n_{r2} \rangle$ *are element-value equal (written as* $n_1 =_{ev} n_2$*) if and only if:*

- n_1 *and* n_2 *both exist and have the same label;*

- *There exists a set M, such that for every pair $(n'_1, n'_2) \in M$, $n'_1 =_{ev} n'_2$, where n'_1, n'_2 are children elements of n_1, n_2, respectively. Every child element of n_1 or n_2 appears in exactly one pair in M.*
- *$(n_1, s) \in \mathcal{V}_1$ if and only if $(n_2, s) \in \mathcal{V}_2$,where s is a simple value.*

Definition 4. (Path-value equality) *Two data element paths p_1 on $T_1 = \langle N_1, \mathcal{P}_1, \mathcal{V}_1, n_{r1}\rangle$ and p_2 on $T_2 = \langle N_2, \mathcal{P}_2, \mathcal{V}_2, n_{r2}\rangle$ are path-value equal (written as $T_1.p_1 =_{pv} T_2.p_2$) if and only if there is a set M' of matching pairs where*

- *For each pair $m' = (n_1, n_2)$ in M', $n_1 \in N_1$, $n_2 \in N_2$, $path(n_1) = p_1$, $path(n_2) = p_2$, and $n_1 =_{ev} n_2$;*
- *All data elements with path p_1 in T_1 and path p_2 in T_2 participate in M', and each such data element participates in only one such pair.*

The definition of functional dependency in XML data needs the definition of so called Generalized Tree Tuple.

Definition 5. (Generalized tree tuple) *A generalized tree tuple of data tree $T = \langle N, \mathcal{P}, \mathcal{V}, n_r\rangle$, with regard to a particular data element n_p (called pivot node), is a tree $t^T_{n_p} = \langle N^t, \mathcal{P}^t, \mathcal{V}^t, n_r\rangle$, where:*

- *$N^t \subseteq N$ is the set of nodes, $n_p \in N^t$;*
- *$\mathcal{P}^t \subseteq \mathcal{P}$ is the set of parent-child edges;*
- *$\mathcal{V}^t \subseteq \mathcal{V}$ is the set of value assignments;*
- *n_r is the same root node in both $t^T_{n_p}$ and T ;*
- *$n \in N^t$ if and only if: 1) n is a descendant or ancestor of n_p in T, or 2) n is a non-repeatable direct descendant of an ancestor of n_p in T ;*
- *$(n_1, n_2) \in \mathcal{P}^t$ if and only if $n_1 \in N^t$, $n_2 \in N^t, (n_1, n_2) \in \mathcal{P}$;*
- *$(n, s) \in \mathcal{V}^t$ if and only if $n \in N^t$, $(n, s) \in \mathcal{V}$.*

A generalized tree tuple is a data tree projected from the original data tree. It has an extra parameter called a pivot node. In contrast with tree tuple defined in [1], which separate sibling nodes with the same path at all hierarchy levels, the generalized tree tuple separate sibling nodes with the same path above the pivot node. See an example generalized tree tuple of tree from Figure 1 in Figure 2. Based on the pivot node, generalized tree tuples can be categorized into tuple classes:

Definition 6. (Tuple class) *A tuple class C^T_p of the data tree T is the set of all generalized tree tuples t^T_n, where $path(n) = p$. Path p is called the pivot path.*

Definition 7. (XML FD) *An XML FD is a triple $\langle C_p, LHS, RHS\rangle$, written as $LHS \rightarrow RHS$ w.r.t. C_p, where C_p denotes a tuple class, LHS is a set of paths (P_{li} , $i = [1, n]$) relative to p, and RHS is a single path (P_r) relative to p.*

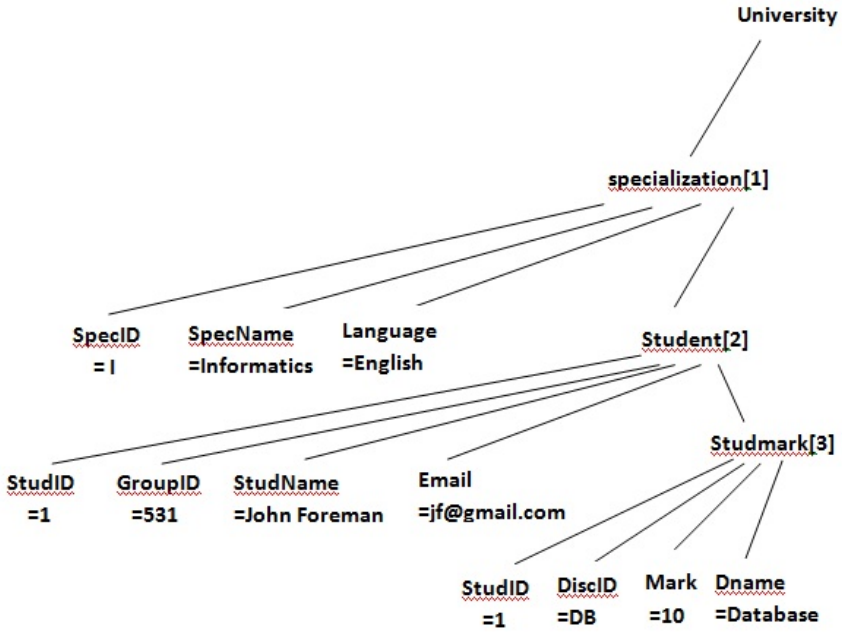


FIGURE 2. Example tree tuple

An XML FD holds on a data tree T (or T satisfies an XML FD) if and only if for any two generalized tree tuples $t_1, t_2 \in C_p$

- $\exists i \in [1,n]$, $t_1.P_{li} =\perp$ or $t_2.P_{li} =\perp$, or

- If $\forall i \in [1,n]$, $t_1.P_{li} =_{pv} t_2.P_{li}$, then $t_1.P_r \neq\perp, t_2.P_r \neq\perp, t_1.P_r =_{pv} t_2.P_r$.

A null value, $\perp$, results from a path that matches no node in the tuple, and $=_{pv}$ is the path-value equality defined in Definition 4.

Example 2. (XML FD) In our running example whenever two disciplines agree on $DiscID$ values, they have the same $DName$. This can be formulated as follows:

$$./DiscID \rightarrow ./DName \text{ w.r.t } C_{Studmark}$$

Another example is:

$$./Student/GroupID \rightarrow ./SpecID \text{ w.r.t } C_{specialization}$$

In our approach we find the XML keys of a given XML document, so we need the next definition:

Definition 8. (XML key) *An XML Key of a data tree T is a pair $\langle C_p, LHS\rangle$, where T satisfies the XML FD $\langle C_p, LHS, ./@key\rangle$.*

Example 3. We have the XML FD: $\langle C_{Student}, ./StudentID, ./Student@key \rangle$, which implies that $\langle C_{Student}, ./StudentID \rangle$ is an XML key.

Tuple classes with repeatable pivot paths are called *essential tuple classes.*

Definition 9. (Interesting XML FD) *An XML FD* $\langle C_p, LHS, RHS \rangle$ *is interesting if it satisfies the following conditions:*

- $RHS \notin LHS$*;*
- C_p *is an essential tuple class;*
- *RHS matches to descendent(s) of the pivot node.*

Definition 10. (XML data redundancy) *A data tree* T *contains a redundancy if and only if* T *satisfies an interesting XML FD* $\langle C_p, LHS, RHS \rangle$*, but does not satisfy the XML Key* $\langle C_p, LHS \rangle$*.*

4. Overview of the Approach

In this section we describe the methodology of a general approach to use FCA to build tools that identify functional dependencies in XML documents. To achieve this, as a first step, we need to define the objects and attributes of interest and create models of XML in terms of FCA context. Our approach is carried out by a sequence of processing steps. The output of each step provides the input to the next step. Every step is illustrated with an example. Our method is supported by a framework named FCAMineXFD. We will now describe each processing step in detail.

4.1. Constructing the Formal Context, the Input to FCA.

In this step the most important issue is how to map the XML document to metamodel entities. Our software can analyze the whole XML document or a *tuple class* C_p given by the path p. Tuple-based XML FD notion proposed in the above section suggests a natural technique for XFD discovery. XML data can be converted into a fully unnested relation, a single relational table, and apply existing FD discovery algorithms directly. Given an XML document, which contains at the beginning the schema of the data, we create generalized tree tuples from it.

Each tree tuple in a tuple class has the same structure, so it has the same number of elements. We use the flat representation which converts the generalized tree tuples into a flat table. Each row in the table corresponds to a tree tuple in the XML tree. In the flat table there are non-leaf and leaf level elements (or attributes) introduced from the tree.

For non-leaf level nodes the associated keys (see Section 3) are used as values.

Applying our experience in detecting functional dependencies in relational databases (see more details in [11]), we use the definitions introduced by

Hereth in [10]. Hereth gives the translation from the relational database into a power context family and based on it he defines the formal context of functional dependencies as follows:

Definition 11. *Let* $\overrightarrow{\mathbb{K}}$ *be a power context family, and let* $m \in M_k$ *be an attribute of the k-th context. Then, the formal context of functional dependencies of* m *with regard to* $\overrightarrow{\mathbb{K}}$ *is defined as* $FD\left(m, \overrightarrow{\mathbb{K}}\right) := \left(m^{I_k} \times m^{I_k}, \{\ 1, 2, ..., k\},\ J\right)$ *with* $((g,h), i) \in J :\Leftrightarrow \pi_i(g) = \pi_i(h)$ *with* $g, h \in m^{I_k}$ *and* $i \in \{\ 1, 2, ..., k\}$.

The π is the relational algebra projection operation. In the next paragraph we will see how we construct this formal context of functional dependencies.

In this step the formal context of functional dependencies for XML data is built, mapping from metamodel entities to FCA objects and attributes.

- Choice of FCA Attributes: *PathEnd/ElementName*
  Due to space considerations we will not specify the whole path to the element (or attribute) names, only the end of the path. FCA attribute names are built from the end of the path to the element: *PathEnd* and element name as follows:
  - for non-leaf level nodes the name of the attribute is constructed as: <ElementName>+"@key" and its value will be the associated key value as specified in Section 3. More elements, which have the same path, will have the same attribute name, but the values will be different.
  - the leaves (actually not the values of the leaves, but the element names of the leaves) of the tree tuple.
- Choice of Objects: the objects are considered to be the tree tuple pairs, actually the tuple pairs of the flat table. The key values associated to non-leaf elements and leaf element's values are used in these tuple pairs.
- Choice of Properties: the mapping between objects and attributes is defined by a binary relation, this incidence relation of the context shows which attributes of this tuple pairs have the same value.

The analyzed XML document may have a large number of tree tuples. Creating the tree tuple pairs, our context table may have a very large number of rows, therefore, we need to clear the concepts of irrelevant entities. We filter the tuple pairs and we leave out those pairs in which there are no common attributes, by an operation called "clarifying the context", which does not alter the conceptual hierarchy.

Example 4. The beginning of the formal context of our running example for tuple class $C_{specialization}$ can be seen in Figure 3. There are only a few

A	B	C	D	E	F	G	H
	specialization/specialization@key	specialization/SpecID	specialization/SpecName	specialization/Language	Student/Student@key	Student/StudID	Student/GroupID
1,I,Informatics,English,2,1,531,John Foreman,Joh...	X	X	X	X	X	X	X
1,I,Informatics,English,2,1,531,John Foreman,Joh...	X	X	X	X			X
1,I,Informatics,English,2,1,531,John Foreman,Joh...	X	X	X	X			X
1,I,Informatics,English,2,1,531,John Foreman,Joh...	X	X	X	X			X
1,I,Informatics,English,2,1,531,John Foreman,Joh...	X	X	X	X			X
1,I,Informatics,English,2,1,531,John Foreman,Joh...	X	X	X	X			X
1,I,Informatics,English,2,1,531,John Foreman,Joh...	X	X	X	X			X
1,I,Informatics,English,2,1,531,John Foreman,Joh...	X	X	X	X			
1,I,Informatics,English,2,1,531,John Foreman,Joh...	X	X	X	X			
1,I,Informatics,English,2,1,531,John Foreman,Joh...	X	X	X	X			
1,I,Informatics,English,2,1,531,John Foreman,Joh...	X	X	X	X			
1,I,Informatics,English,2,1,531,John Foreman,Joh...	X	X	X	X			
1,I,Informatics,English,2,1,531,John Foreman,Joh...	X	X	X	X			
1,I,Informatics,English,2,1,531,John Foreman,Joh...	X	X	X	X			
1,I,Informatics,English,2,1,531,John Foreman,Joh...	X	X	X	X			
1,I,Informatics,English,2,1,531,John Foreman,Joh...	X	X	X	X			
1,I,Informatics,English,2,1,531,John Foreman,Joh...	X	X	X	X			
1,I,Informatics,English,2,1,531,John Foreman,Joh...	X	X	X	X			
1,I,Informatics,English,2,1,531,John Foreman,Joh...				X			
1,I,Informatics,English,2,1,531,John Foreman,Joh...				X			
1,I,Informatics,English,2,1,531,John Foreman,Joh...				X			
1,I,Informatics,English,2,1,531,John Foreman,Joh...				X			

FIGURE 3. Beginning of the Formal Context of functional dependencies for tuple class $C_{specialization}$

columns of it in the image, due to space considerations. We can see the attributes as column names, like *Student/Student@key* (for non-leaf element), *specialization/SpecName* (for leaf element). Rows contain the tuple pairs, only the beginning of them can be seen. If tuple pairs has the same value for an attribute, then X appears in the context table. This file will be the input for the next step.

4.2. **Creating the Concept Lattice.** Once the objects and attributes of the context are defined, we run the Concept Explorer (ConExp) [21] engine to generate the concepts and create the concept lattice. The main output produced by FCA is the concept lattice.

Example 5. The concept lattice for the formal context of functional dependencies for XML data constructed in previous step for tuple class $C_{specialization}$ can be seen in Figure 4.

4.3. **Processing the Output of FCA.** A concept lattice consists of the set of concepts of a formal context and the subconcept-superconcept relation between the concepts, see [8]. Every circle in Figure 4 represents a formal concept. Each concept is a tuple of a set of objects and a set of common attributes, but only the attributes are listed. An edge connects two concepts if one implies the other directly. Each link connecting two concepts represents the transitive subconcept-superconcept relation between them. The top concept has all formal objects in its extension. The bottom concept has all formal attributes in its intension.

Example 6. In Figure 4 node labeled with $Student/GroupID$ is on upward path from node labeled by $Student/StudID$, $Student/StudName$, $Student/$ $Student@key$, $Student/Email$. In FCA language, concept with label $Student/$

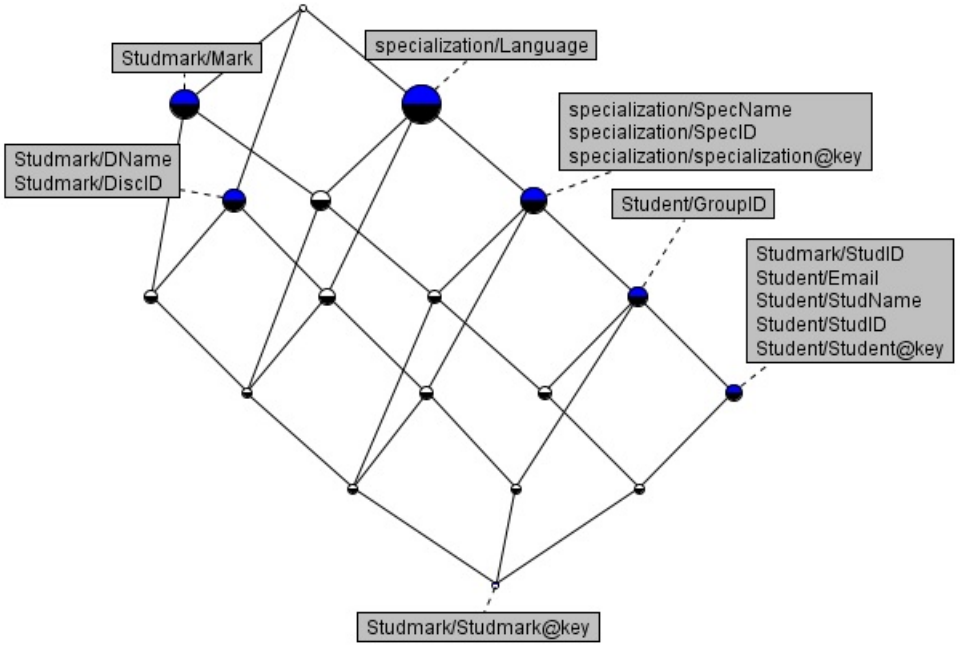


FIGURE 4. Concept Lattice of functional dependencies' Formal Context for tuple class $C_{specialization}$

$StudID$, $Student/StudName$, $Student/Student@key$, $Student/Email$ implies concept with label $Student/GroupID$.

4.4. **Mining XFDs according to the concept hierarchy.** In this step, we examine the candidate concepts resulting from the previous steps and use them to explore XFDs. Once the lattice is constructed, we can interpret each concept and generate the list of all functional dependencies.

The relationship between FDs in databases and implications in FCA was pointed out in [8]: a FD $X \rightarrow Y$ holds in a relation r over R iff the implication $X \rightarrow Y$ holds in the context (G, R, I) where $G = \{(t_1, t_2) | t_1, t_2 \in r, t_1 \neq t_2\}$ and $\forall A \in R$, $(t_1, t_2) I A \Leftrightarrow t_1[A] = t_2[A]$.

This means that objects of the context are couples of tuples and each object intent is the agree set of this couple. Thus, the implications in this lattice corresponds to functional dependencies in XML.

Example 7. Analyzing the Conceptual Lattice obtained for tuple class $C_{Student}$ (Figure 4) we can detect functional dependencies like:

$\langle C_{Student}, ./StudID, ./GroupID\rangle$

$\langle C_{Student}, ./Student@key, ./GroupID\rangle$

$\langle C_{Student}, ./StudName, ./GroupID\rangle$

$\langle C_{Student}, ./Email, ./GroupID\rangle$

In the lattice we list only the attributes, these are relevant for our analysis. Let there be a concept, labeled by A, B and a second concept labeled by C. A, B and C are FCA attributes. Let concept labeled by A, B be the subconcept of concept labeled by C. Therefore tuple pairs of concept labeled by A, B have the same values for attributes A, B, but for attribute C too. Tuple pairs of concept labeled by C do not have the same values for attribute A, nor for B, but have the same value for attribute C. Tuple pairs of every subconcept of concept labeled by A, B have the same values for attributes A, B. The labeling of the lattice is simplified by putting each attribute only once, at the highest level. We analyze attributes A and B. If we have only $A \to B$, then A would be a subconcept of B. If only $B \to A$ holds then B should be a subconcept of A. We have $A \to B$ and $B \to A$, that's why they come side by side in the lattice. So attributes from a concept imply each other.

Example 8. In concept node with label $specialization/specialization@key$, $specialization/SpecID$, $specialization/SpecName$ the associated objects are tree tuple pairs, where the values for $specialization/SpecID$ are the same. So we have the next XML FDs:

$\langle C_{specialization}, ./SpecID, ./SpecName\rangle$

$\langle C_{specialization}, ./SpecID, ./specialization@key\rangle$

$\langle C_{specialization}, ./specialization@key, ./SpecID\rangle$

$\langle C_{specialization}, ./specialization@key, ./SpecName\rangle$

$\langle C_{specialization}, ./SpecName, ./specialization@key\rangle$

$\langle C_{specialization}, ./SpecName, ./SpecID\rangle$

Software FCAMineXFD found many functional dependencies. A part of these XML FD-s are in Figure 5.

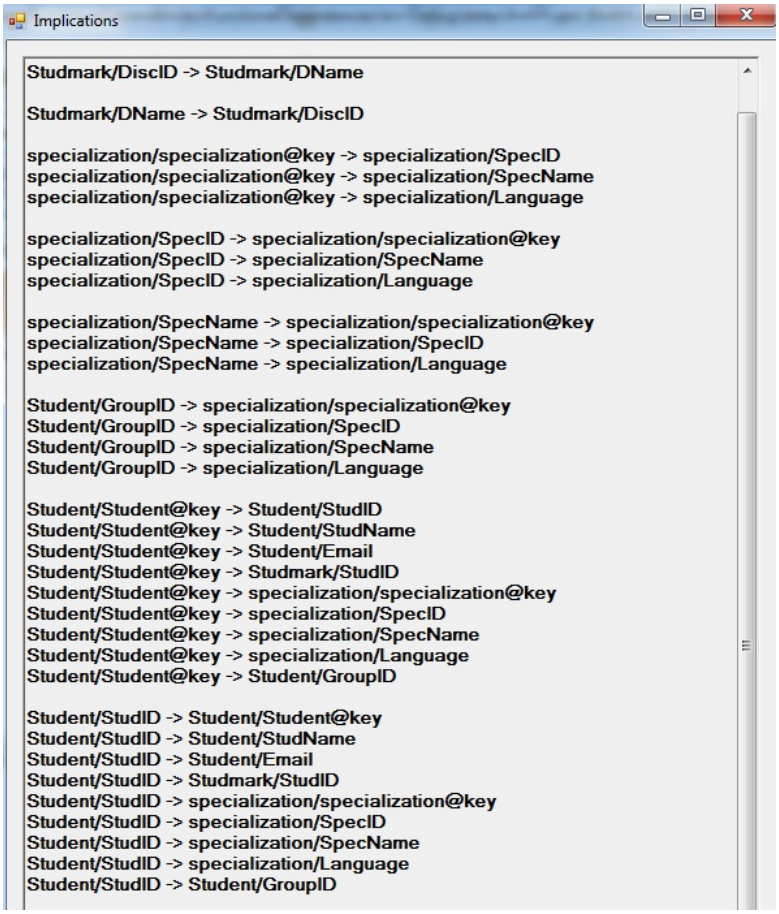


FIGURE 5. Some of the functional dependencies in tuple class $C_{specialization}$

Example 9. In the concept lattice for the XML document of Example 1 we can see the hierarchy of the analyzed data. The node labeled by *specialization/Language* is on a higher level, than node labeled by *specialization/SpecName*. The *specialization* node with three attributes is a subconcept of node labeled *specialization/Language*. The *Student* node in XML is child of *specialization* node. In the lattice, the node labeled with the key of Student, is subconcept of *specialization* node, so the hierarchy is visible. These are 1:n relationships, from *Specialization* to *Group*, from *Group* to *Students*, from *Students* to *Studmark*.

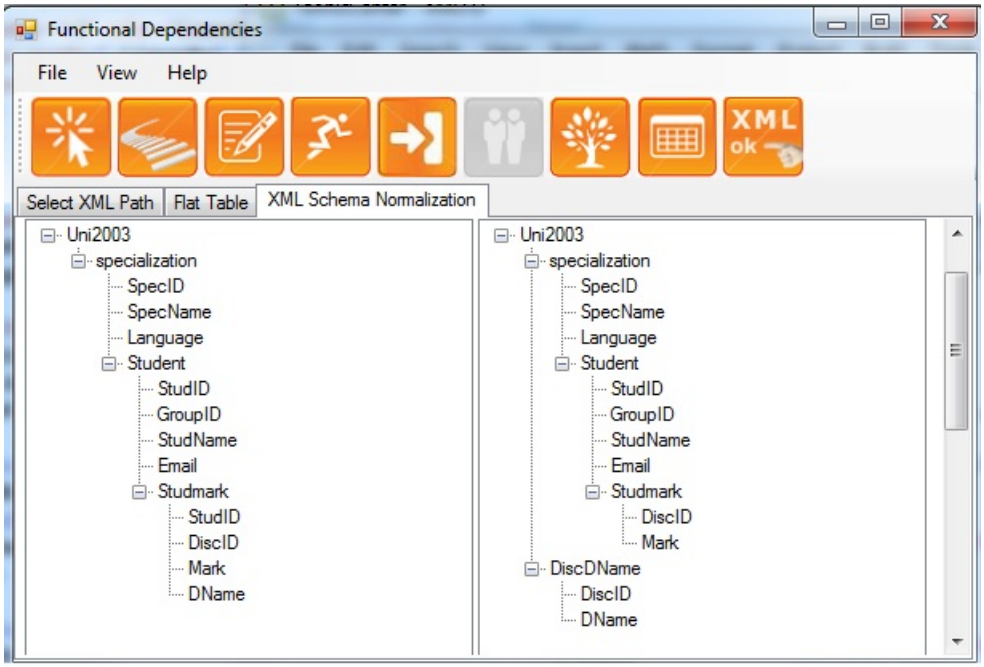


FIGURE 6. The proposed new scheme of University XML data

The information about *Disciplines* is on the other side of the lattice. *Disciplines* are in n:m relationship with *Students*, linked by *Studmark* node in this case.

Therefore, we say that FCA can serve as a guideline for dependency mining.

4.5. **Finding XML keys.** The implications found by FCAMineXFD contain some FDs with RHS as ./@key values. These can be used to detect the keys in XML.

Example 10. In tuple class $C_{Student}$ we have XML FD: $\langle C_{Student}, ./StudentID, ./Student@key\rangle$, which implies that $\langle C_{Student}, ./StudentID\rangle$ is an XML key.

Example 11. Let us look at XML FD-s of Example 8. There are two FDs with RHS as $./specialization@key$ in tuple class $C_{specialization}$, so the detected XML keys are: $\langle C_{specialization}, ./SpecID\rangle$, $\langle C_{specialization}, ./SpecName\rangle$.

4.6. **Detecting XML data redundancy.** Having the set of functional dependencies for XML data in a tuple class, we can detect interesting functional dependencies. In essential tuple class $C_{Studmark}$, the XML FD $\langle C_{Studmark},$

$./DiscID, ./DName\rangle$ found by the software in Figure 5 is an interesting FD, but $\langle C_{Studmark}, ./DiscID\rangle$ is not an XML key. So it is a data redundancy.

The same reason applies for XML FD $\langle C_{Studmark}, ./DName, ./DiscID\rangle$.

4.7. **Normalization.** Given the set of dependencies discovered by our tool, we adopt the normalization algorithm of [22] to convert one XML schema into a correct one. See the resulting scheme in Figure 6.

5. Conclusion and Future Work

This paper introduces an approach for mining functional dependencies in XML documents based on FCA. We proposed a framework to analyze XML documents using Concept Analysis. Based on the flat representation of XML, we constructed the concept lattice. We analyzed the resulted concepts, which allowed us to discover a number of interesting dependencies. Our framework offers an interactive visualization for dependency exploration. Taking in consideration our preliminary results, we believe that FCA is a promising technique in XML database design too. We had also previously used FCA to explore functional dependencies in relational databases, see more details in [11]. In this paper, we complemented the information with XML design exploration.

Given the set of dependencies discovered by our tool, we propose a correct XML schema. We have started to use our approach on several case studies. We plan to develop our own FCA tool because Conexp is limited w.r.t. number of rows in the formal context.

6. Acknowledgement.

The author Viorica Varga has been fully supported by Romanian Ministry of Education in the frame Grant CNCSIS PCCE-55/2008.

References

[1] Arenas, M., Libkin, L.: *A normal form for XML documents.* TODS 29(1), pp. 195-232 (2004)

[2] Arenas, M., Libkin, L.: *An information-theoretic approach to normal forms for relational and XML data.* JACM 52(2), pp. 246-283 (2005)

[3] Arenas, M., Libkin, L., Fan, W.: *On the Complexity of Verifying Consistency of XML Specifications.* SIAM J. Comput. 38(3), pp. 841-880 (2008)

[4] Baixeries, J.: *A formal concept analysis framework to mine functional dependencies.* Proceedings of Mathematical Methods for Learning (2004)

[5] Buneman, P., Davidson, S., Fan, W., Hara, C., Tan, W.-C.: *Keys for XML.* In: Proc. WWW, 201-210. Hong Kong, China (2001)

[6] Chen, Y., Davidson, S., Hara, C., Zheng, Y.: *RRXS:redundancy reducing XML storage in relations.* In: VLDB, pp. 189-200 (2003)

[7] Embley, D.W., Mok, W.Y.: *Developing XML documents with guaranteed "good" properties*. In: ER 2001, 20th International Conference on Conceptual Modeling, pp. 426-441 (2001)
[8] Ganter, B., Wille, R.: *Formal Concept Analysis. Mathematical Foundations*. Springer (1999)
[9] Hartmann, S.: *Axiomatising functional dependencies for XML with frequencies*. In: FOIKS, pp. 159-178 (2006)
[10] Hereth, J.: *Relational Scaling and Databases*. Proceedings of the 10th International Conference on Conceptual Structures: Integration and Interfaces LNCS 2393, Springer Verlag, pp. 62-76 (2002)
[11] Janosi Rancz, K.T., Varga, V., Puskas, J.: *A Software Tool for Data Analysis Based on Formal Concept Analysis*. Studia Univ. Babeş-Bolyai, Informatica, 53, 2, pp. 67-78 (2008)
[12] Katalin Tunde Janosi-Rancz, Viorica Varga, and Timea Nagy: *Detecting XML Functional Dependencies through Formal Concept Analysis*, 14th East European Conference on Advances in Databases and Information Systems (ADBIS), Novi Sad, Serbia, LNCS 6295, pp. 595-598 (2010).
[13] Lee, M., Ling, T., Low, W.L.: *Designing functional dependencies for XML*. In: EDBT Conference, pp. 124-141 (2002)
[14] Lopes, S., Petit, J-M., Lakhal, L.: *Functional and approximate dependency mining: database and FCA points of view*. Special issue of Journal of Experimental and Theoretical Artificial Intelligence (JETAI) on Concept Lattices for KDD, 14(2-3): pp. 93-114, Taylor and Francis (2002)
[15] Lopes, S., Petit, J-M., Lakhal, L.: *Efficient Discovery of Functional Dependencies and Armstrong Relations*. Proceedings of the 7th International Conference on Extending Database Technology (EDBT), Konstanz, Germany (2000)
[16] Schewe, K.D.: *Redundancy, dependencies and normal forms for XML databases*, In: ADC, pp. 7-16 (2005)
[17] Vincent, M. W., Liu, J., Mohania, M.: *On the Equivalence between FDs in XML and FDs in Relations*. Acta Informatica 44(3-4), pp. 207-247 (2007)
[18] Vincent, M. W., Liu, J., Liu, C.: *Strong functional dependencies and their application to normal forms in XML*. ACM TODS, 29(3): pp. 445-462 (2004)
[19] Wang, J.,Topor, R.: *Removing XML data redundancies using functional and equality-generating dependencies*, In: ADC, pp. 65-74 (2005)
[20] W3C. XML Schema, http://www.w3.org/TR/xmlschema-0/ (2004)
[21] Serhiy Yevtushenko, A.: *System of data analysis "Concept Explorer"*. (In Russian). Proceedings of the 7th National Conference on Artificial Intelligence KII-2000, Russia, pp. 127-134 (2000)
[22] Yu, C., Jagadish, H. V.: *XML schema refinement through redundancy detection and normalization*. VLDB J. 17(2): pp. 203-223 (2008)

Hungarian University of Transylvania, Tirgu Mures, Romania
E-mail address: tsuto@ms.sapientia.ro

Babes-Bolyai University, Cluj, Romania
E-mail address: ivarga@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVII**, Number 3, 2012

ANALYSING THE USAGE OF PULSE PORTAL WITH FORMAL CONCEPT ANALYSIS

SANDA DRAGOŞ AND CHRISTIAN SĂCĂREA

Abstract. This paper aims to present an analysis of web usage using Conceptual Knowledge Processing and Representation tools, mainly Formal Concept Analysis and the knowledge management system TOSCANAJ. We describe several knowledge maps in form of conceptual hierarchies in order to highlight how these tools might be used for analysing access data in the PULSE system.

1. Introduction

Formal Concept Analysis (FCA) [3] is a powerful tool in order to represent and process knowledge. We have been mainly interested in evaluating and representing web usage data, which are usually interpreted using web analytics or data mining techniques.

By this approach, data have been gathered and then analysed by the knowledge management system TOSCANA. Using this system, concepts reflecting web usage have been built and then represented in conceptual hierarchies. These hierarchies are then used as knowledge maps allowing us to analyse and hence to build valuable judgements over our data.

To the best of our knowledge FCA was not used for interpreting web site usage data.

2. Formal Concept Analysis

Formal Concept Analysis is a mathematical theory widely used in data analysis. The basic structure is the *formal context*, which exploits the fact that data is quite often represented as objects and attributes. Attributes

Received by the editors: October 6, 2012.

2010 *Mathematics Subject Classification.* 68T30 Knowledge representation, 68P20 Information storage and retrieval.

1998 *CR Categories and Descriptors.* H.3.5 [Information Systems]: Information Storage and Retrieval – *On-line Information Services*.

Key words and phrases. formal concept analysis, web analytics.

might be either simple (attributes have binary values, YES/NO) or many-valued (attributes have values). Objects are linked to their attributes by an incidence relation. Because of the specificity of our data, we have considered manyvalued contexts. These are then scaled, i.e., transformed by means of a conceptual analysis into simple, binary contexts. For these contexts, concept are built and displayed in conceptual hierarchies, i.e., order diagrams.

Formally, a formal context is a triple (G, M, I), where G is a set of objects, M is a set of attributes, and I binary relation $I \subseteq G \times M$, called the *incidence relation*. Thus, gIm is read *the object g has attribute m*. A fininte context can be represented as a cross-table, the rows labeled by object names, the columns by attribute name and the incidence relation is represented by crosses.

A manyvalued context is defined as (G, M, W, I), G being the set of object, M the set of attributes, W the set of attribute values, and I a ternary relation linking object, attributes and their values, also called incidence relation. Here $(g, m, w) \in I$ means that *object g has attribute m with value w*. However, the data needs to be interpreted/converted from many-valued attributes into a single value attribute. This process is called conceptual scaling by Ganter and Wille [3].

FCA is a formalization of the classical, philosophical understanding of concepts and their importance for knowledge. We define two derivation operators, which proves to be a Galois connection between the power sets of G and M (see [3]). These operators are defined as follows: Let (G, M, I) be a formal context and $A \subseteq G, B \subseteq M$ be subsets of objects and attributes, respectively.

$$A' := \{m \in M \mid \ gIm, \ \forall g \in A\}.$$
$$B' := \{g \in G \mid \ gIm, \ \forall m \in B\}.$$

A formal concept is a pair (A, B) of sets, $A \subseteq G$, $B \subseteq M$ with $A' := B, B' := A$. A descriptive interpretation is the following: All commonly shared attributes of all objects in A belong to B and all attributes in B are shared by all objects in A. Not so obvious is the fact that (A, B) is maximal with this property. Using the cross-table representation of data, a concept is a maximal rectangle of crosses, property which is directly derived from the above definition. Concepts might also be understood as basic units of knowledge, since they reflect a basic structure of data clustering.

We denote the set of all concepts by $\mathfrak{B}(G, M, I)$. This set is structured by an order relation, called the *subconcept-superconcept relationship*. This is a specialization-generalization relationship for concepts. We say that (A, B) is a *subconcept* of (C, D) (or a specialization of it) if and only if A is a subset of C (which is of course equivalent to D being a subset of B). The concept (C, D) is called superconcept (or generalization) of (A, B). This relation is an order relation over the set of all concepts:

$$(A, B) \leq (C, D) :\Leftrightarrow A \subseteq C (\Leftrightarrow D \subseteq B).$$

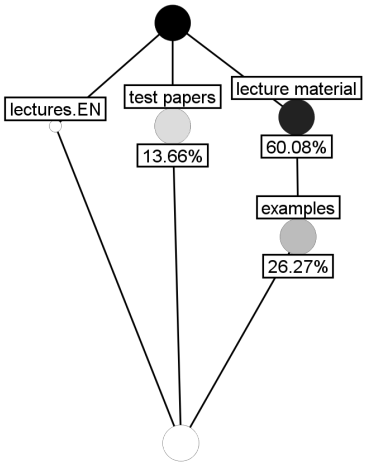


FIGURE 1. Example of a Hasse diagram representing conceptual hierarchies

The ordered set $(\mathfrak{B}(G, M, I), \leq)$ is a complete lattice, called *conceptual hierarchy*, or *concept lattice* (for more information we refer to [3]). The conceptual hierarchy is then used as a knowledge map, since it represents the encoded knowledge in the data under analysis.

These conceptual hierarchies can be graphically represented by a order diagrams with a simple and easy to read structure and labelling as depicted in Figure 1.

How do we read such a diagram? Every node is a (formal) concept, i.e., a maximal collection of objects and common attributes. The set of objects is called extent, that of attributes intent. The same object or attribute might be part of different concepts, thus establishing a hierarchy which enables navigation. To illustrate this of Figure 6, the node labelled *'examples'* lies below the node labelled *'lecture material'*, that means that *'examples'* is a subconcept of *'lecture material'*. The percentages below every node represent the access percentages.

R. Wille considered ([4]) that knowledge can be represented in conceptual landscapes, hence navigation must be provided from one concept to another. The conceptual hierarchies, as being described in [5] and [6], act both as knowledge maps but also as a guide through the data set under consideration. They not only represent knowledge, but also make possible knowledge acquisition, knowledge processing and a special view, called *conceptual*. Conceptual means that the queries we make, the navigation and zooming into different parts of our knowledge landscapes are driven by a set of concepts, a set of

clearly structured ideas which can also be read off from the hierarchies under consideration.

This view has been implemented in the knowledge management system ToscanaJ [1]. This system comprises three different tools, Elba, Siena and Toscana itself.

Data is usually stored in a database to which Elba connects. This database is interpreted as a many-valued context, hence every many-valued attribute has to be scaled, according to some rules, called conceptual scaling. The process of scaling once finished (for all, or just for some attributes of interest), a conceptual schema is provided. Siena just consideres the formal context output of conceptual scaling for further actions. The conceptual schema is then opened by ToscanaJ, which is the visualisation tool of this suite.

In our approach, we have used ToscanaJ [1] to build the conceptual hierarchis and then to browse the formed conceptual patterns over the data set. Elba [7] was used to build the scales over the database containing the PULSE accesses and to export schemas to be explored with Toscana. Browsing with Toscana offers the opportunity to use the available diagrams according to specific needs; it is possible to aggregate the diagrams in order to obtain results such as proof of the existence of patterns in attributes correlation. Different scenarios can be formed using only a small subset of the diagrams.

3. Formal Concept Analysis on web usage data

The web site used for collecting the usage/access data is an e-learning portal called PULSE [2]. The analysis was done only on the data collected from the two months of the last academic semester (i.e. April and May of 2012).

We have started our analysis by determining who is using PULSE and in what proportion. As depicted in Figure 2, students use this portal in proportion of almost 40%, which in this case means 8334 accesses. The teacher using PULSE accessed it 434 times (=2.08%) The rest of 58.07% accesses appear under the label *'no login'* and take place just before the login phase when no login name can be recorded. Those 12142 accesses are made in the pre-login phase by both students and teacher (as all students and the teacher accessed the login page), but also by other individuals which did not have a PULSE account or landed on the PULSE login page accidentally.

The diagram can be also configured to show the number of accesses instead of login distributions. Thus, we can also determine the exact number of accesses as described above.

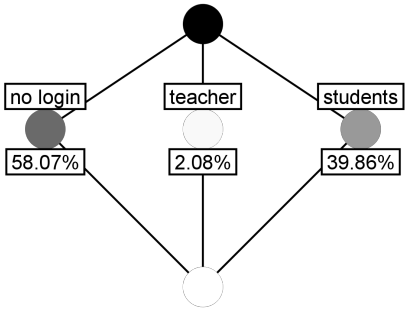


FIGURE 2. Who is using PULSE? Login distribution.

We continued our investigation by determining which students are using PULSE. Therefore, we generated the diagram from Figure 3 which divides student accesses by their accessed year of study.

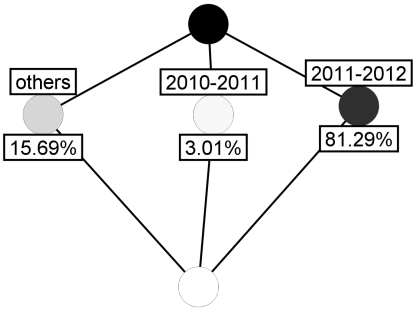


FIGURE 3. Which students? Years of study.

The results show that students revisit PULSE for checking the information taught in previous years. There were 6775 student accesses for currently taught subjects, 215 student accesses for subjects taught last, and 1308 student accesses for subjects taught in other previous years. We mention that the same subject on PULSE has a different content on different years of study as the teaching data is updated yearly, and each year a new set of students would have accounts created for the specific subjects. One student may have a single account but he/she can navigate through the different subject and years of study on which he/she has access on PULSE.

Figure 4 shows the access distribution of PULSE students on different subjects. As in the recorded semester (e.g. from February to June 2012), the subject studied was SO1 (i.e., Operating Systems), the percentage of almost 85% (=7015 accesses) is justifiable. What is interesting to observe here is that

the other subjects, although studied in the previous semester are still revisited. This is surprising because students were examined on those subjects and they still return to revisit the information posted there. The 15.73%=1311 accesses were generated by visiting the pages before the subject selection phase.

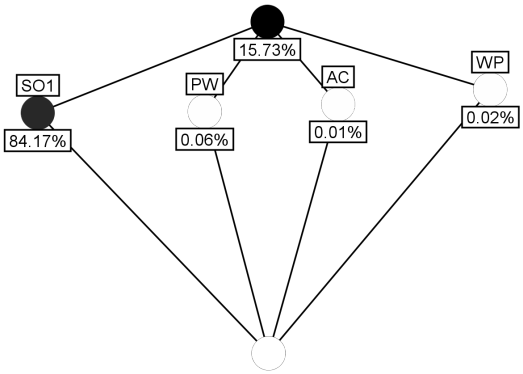


FIGURE 4. Teaching subjects for students

Over the considered period of time, the teacher accessed only the current subject as depicted in Figure 5. The 5.07% accesses are the ones recorded in each visit prior to the subject selection.

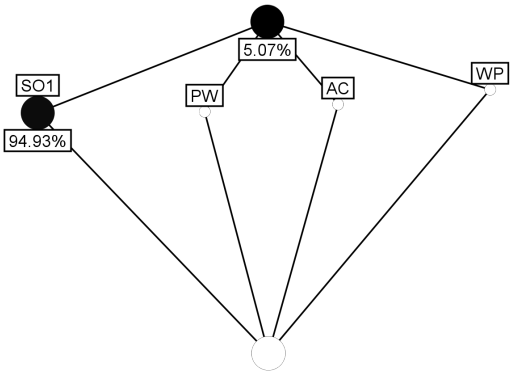


FIGURE 5. Teaching subjects for teacher

As a next step we wanted to check how well PULSE performs for the task that it was designed for. We wanted to see if students access the information provided. Figure 6 shows these results on different levels of importance. The distributions presented here confirmed our expectations. PULSE was designed as support instrument for laboratories and lectures. These classes have

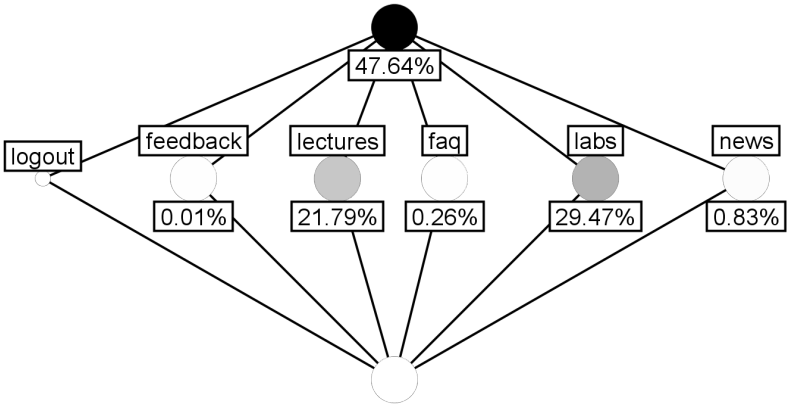


FIGURE 6. student PULSE accesses by classes

obtained the highest score. The 47.64% accesses were made on the PULSE *'homepage'*. This page loads just after the login phase and contains general information for students, such as:

- the name of the authenticated person;
- the group of the student;
- notifications/announcements from the teacher;
- for each laboratory specific information such as:
  - the week within the semester and corresponding calendar dates;
  - the name of the concept studied;
  - the corresponding assignment reference;
  - the mark if the assignment was handed;
  - and the attendance status
- lab activity (i.e., average score and the total number of attendances)
- the marks for the practical and written exam (at the end of the semester) as well as the final mark.

Therefore, this page is visited very often. The other facilities offered by PULSE are also presented in Figure 6. Students accessed 69 times the *'news'* page which contains all notifications/announcements made by the teacher for that specific subject. The last announcement is always posted also on the *'homepage'*. The *'faq'* (Frequently Asked Questions) page was accessed 22 times, while only one student access was made to send a *feedback*. A very insightful result is that students do not use the *logout* button.

ToscanaJ allows aggregating these diagrams and navigating from one diagram to another. For instance, if we select the node *'lectures'* from Figure 6, we can zoom into this node, in order to evaluate access distributions for lectures. The result is displayed in Figure 7a. In the same way, zooming into node *'labs'*, the result of access distribution is displayed in Figure 7b. Hence, by combining different scales (the above mentioned diagrams), we can build several scenarios of browsing the knowledge content of our database.

This browsing is reversible, at any point we may return and choose another scenario, highlighting other connections between our data. These scenarios might be understood as changing the point of view of our analysis.

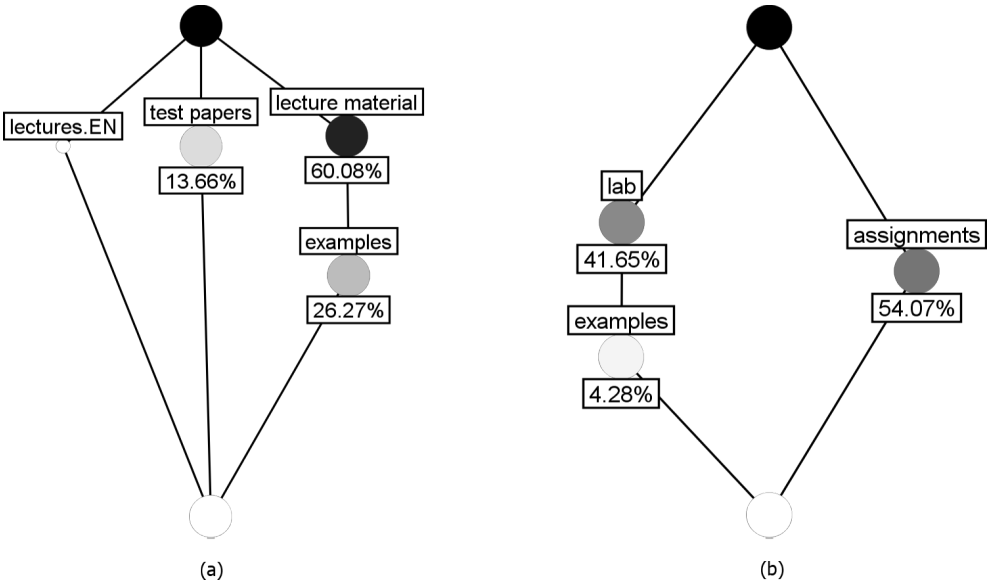


FIGURE 7. (a) access distribution for lecture related material, (b) access distribution for labs

As shown in Figure 7a, the most visited lecture related pages are those containing the theoretical support which consists *'lecture material'* and more *'examples'*, which means $60.08\% + 26.27\% = 86.35\%$. Then, there are the test papers during lectures and their results (including statistics and explanation how their marks will help the student) are presented under *'test papers'*. The detailed explanations on the solved test papers and statistics on the proportion in which the subjects of the test paper were solved by all students are placed together with the examples (i.e., under *'example'*. The lecture material is presented in Romanian. However, there are also English lectures. The

considered semester there were no English lectures, therefore there were no accesses for the node *'lectures.EN'*.

Figure 7b shows the detailed view on *'labs'*. Similar with lectures, here there is the technical support presenting the required concepts and *'examples'*. To better understand the concepts, students are given *'assignments'*.

PULSE facilities offered for teachers are presented in Figure 8.

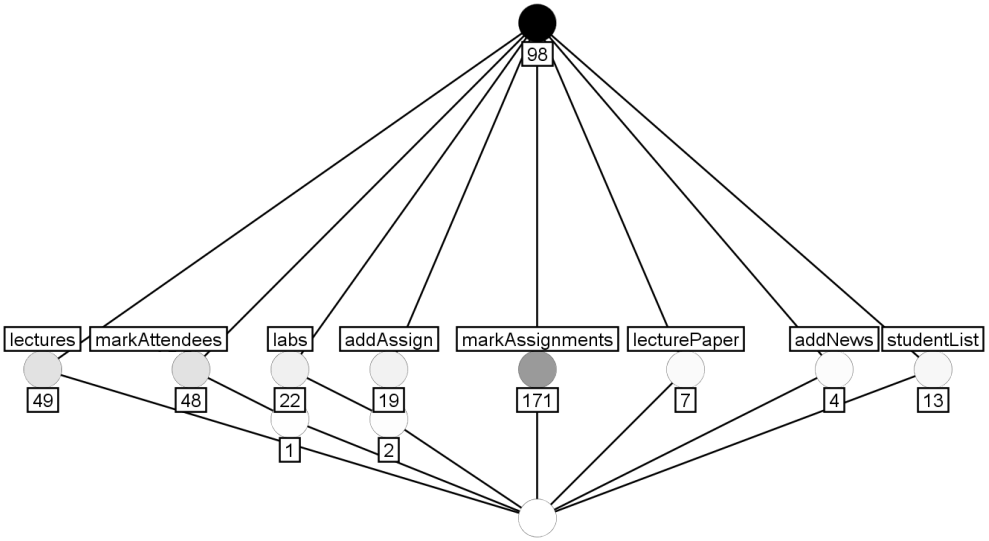


FIGURE 8. PULSE for teachers

Those facilities, in their most accessed order are:

- 39.40% *markAssignments* - is to mark student scores (the most used PULSE facility);
- 11.29% *lectures* - check the course support;
- 11.06% *markAttendees* - mark student attendancies;
- 5.07% *labs* - check lab support;
- 4.38% *addAssign* - assign random tasks for students;
- 3.00% *studentList* - list all students with their marks, attendances and final scores;
- 1.61% *lecturePaper* - list all students which have lecture paper marks;
- 0.92% *addNews* - add notifications/announcements.

Combining two or more conceptual hierarchy in one diagram is also possible using *nested-line diagrams*. These diagrams are obtained by partitioning

the attribute set in two or more subsets and then considering the direct product of the resulting conceptual sub-hierarchies. The original conceptual hierarchy is embedded in this direct product. This representation is particularly suitable if we would like to analyse the behaviour of our data with respect to a given collection of attributes. The following example depicted in Figure 9 shows a nested line diagram obtained by combining the months in the considered period and PULSE actors.

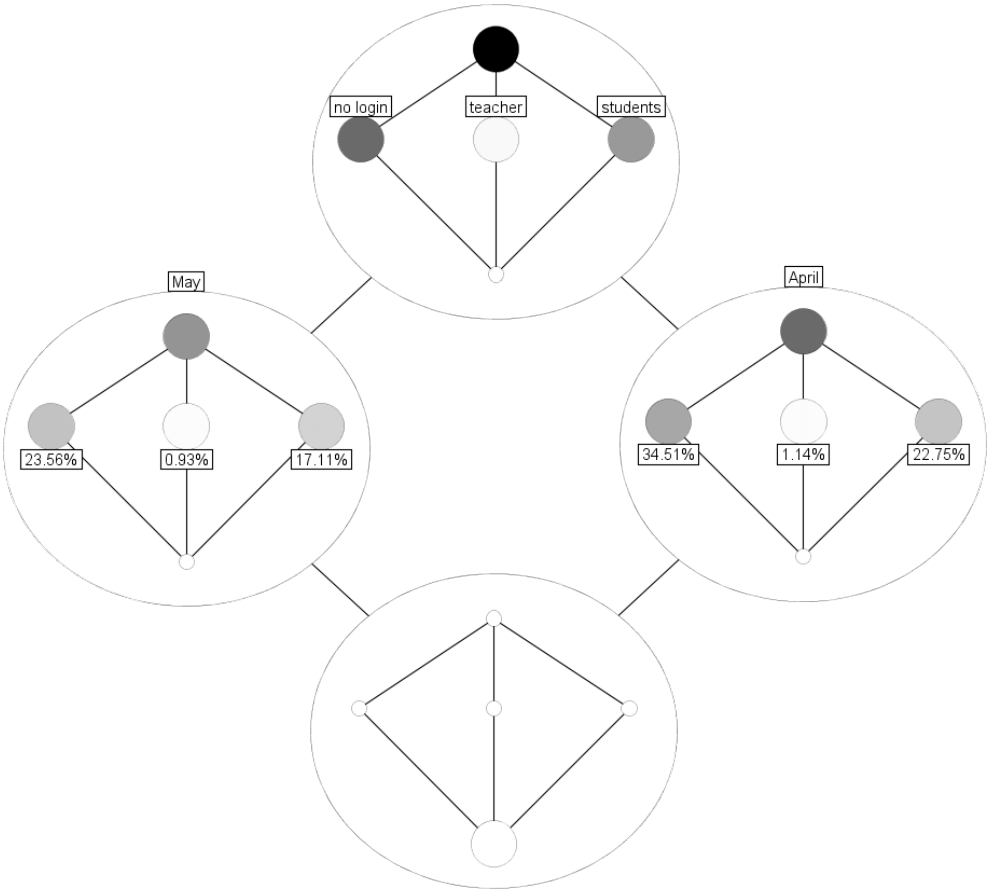


FIGURE 9. Nested diagram: months & login

4. CONCLUSIONS

The above considerations are reflecting a new approach into analysing web usage data. FCA allows more flexibility in analysing data by using predefined

concepts. Using the FCA diagrams one may navigate and interpret data from multiple different perspectives. By comparison, web analytics offers more limited perspective by using a set of metrics which can be interpreted based on their values.

We are now mainly interested in connections between our data and how the information gathered by the PULSE portal are linked and connected. Formal Concept Analysis seems to be a proper way to perform such analysis. Due to lack of space, we have not been able to describe several analysis scenarios, but more the main approach and analysis method, which have been, for the purpose of this paper of importance.

For a further research, we would like to investigate some of these relevant scenarios and to use conceptual logic, in form of attribute exploration and association rule mining might be helpful.

A possible development of this research might also be into the triadic setting of Triadic Formal Concept Analysis, but this will be presented in our future papers.

References

[1] P. Becker, J. Hereth, and G. Stumme, *Toscanaj - an open source tool for qualitative data analysis*, (2002).

[2] S. Dragos, *PULSE Extended*, in The Fourth International Conference on Internet and Web Applications and Services, Venice/Mestre, Italy, May 2009, IEEE Computer Society, pp. 510–515.

[3] B. Ganter and R. Wille, *Formal concept analysis: mathematical foundations*, Springer Verlag, 1999.

[4] R. Wille, *Conceptual landscapes of knowledge: a pragmatic paradigm for knowledge processing*, in International Symposium on Knowledge Representation, Use, and Storage Efficiency, G. Mineau and A. Fall, eds., Vancouver, 1997, Simon Fraser University, pp. 2–13.

[5] ———, *Begriffliche wissensverarbeitung: Theorie und praxis*, Informatik Spektrum, (2000).

[6] ———, *Methods of conceptual knowledge processing*, in the 4th International Conference ICFCA, Springer Verlag, 2006, pp. 1–29.

[7] B. Wormuth, *Elba user manual*. Published online, 2004. http://kvo.uow.edu.au/kvopapers/Elba_User_Manual.pdf.

UBB Cluj-Napoca
E-mail address: sanda@cs.ubbcluj.ro, csacarea@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVII**, Number 3, 2012

A COLLABORATIVE EVOLUTIONARY APPROACH TO RESOURCE-CONSTRAINED PROJECT SCHEDULING

ANCA ANDREICA, CAMELIA CHIRA

Abstract. Resource-constrained project scheduling is an NP-hard optimization problem focusing on the task of time-dependent resource allocation for a project. The current paper presents the application of a geometric collaborative evolutionary algorithm to this problem. Important features of the evolutionary model include population topology, asynchronous search and adaptive selection/recombination strategy. Each individual has an agent-inspired behaviour in the sense that communication with other individuals is possible and facilitates the selection of a mate for recombination. The evolving population has a geometrical structure and is furthermore organized in dynamic societies with different strategies for recombination. Numerical experiments are performed for several project instances and results emphasize a good performance of the geometric collaborative evolutionary model.

1. Introduction

The Resource-Constrained Project Scheduling Problem (RCPSP) is of great importance in a large number of application areas including construction and civil engineering, manufacturing, production planning, logistics and project management. RCPSP requires the allocation of limited resources to dependent activities over time, such that the makespan of the project is minimized. The challenges associated with RCPSP relate to complex resource constraints as well as activity dependencies. It has been shown that RCPSP belongs to the class of NP-hard optimization problems [1] which means that exact solutions can not be found in polynomial time by running an algorithm. Therefore, there is a high interest in developing good approximation methods to address RCPSP with the aim of finding near-optimal (or optimal) solutions using limited resources. Inspired by the process of natural evolution, genetic

Received by the editors: October 8, 2012.

2010 *Mathematics Subject Classification.* 68-02.

1998 *CR Categories and Descriptors.* I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search – *Heuristic methods*.

Key words and phrases. evolutionary computation, resource-constrained project scheduling, crossover, permutation based encoding, collaborative evolutionary algorithms.

algorithms represent good approximation methods for solving problems belonging to this class. Bio-inspired heuristic approaches to RCPSP include a genetic algorithm with a new permutation of priority-based encoding scheme [13], a permutation-based elitist genetic algorithm [8], an algorithm based on a priority value encoding scheme [12] and a hybrid genetic algorithm [11].

The evolutionary framework used in this paper is the Geometric Collaborative Evolutionary (GCE) model proposed in [2]. In this model, the population has a geometrical structure given by the circular placement of individuals on layers according to their fitness. Furthermore, the agent-inspired component of GCE leads to the collaboration of individuals which facilitates selection and recombination. The search process is asynchronous allowing the improvement and replacement of individuals within the same generation. The three main strategies for recombination are supported by three co-evolving and dynamic subpopulations or societies of individuals (i.e. *local*, *far* and *global*) having different policies for mate selection and recombination. Population dynamics emerges through the recombination of individuals from different societies and a dominance principle. Co-evolution of societies enables a useful balance between search diversification and intensification. Some individuals are specialized for local search facilitating exploitation while other individuals focus on global search. The GCE model reports promising results for various difficult unimodal and multimodal real-valued functions with many dimensions [2, 5] and for the problem of evolving Cellular Automata rules [3] in which the real and, respectively, binary representation have been used.

In the current paper, the GCE model is adapted for the application to RCPSP based on the permutation representation. Computational experiments are presented for RCPSP based on several project instances and the results of the RCPSP-customized GCE algorithm are compared to that of a standard evolutionary algorithm.

The paper is organized as follows: Section 2 presents the RCPSP addressed in this paper, Sectiom 3 describes the GCE model, Section 4 presents the GCE-based approach to RCPSP, Section 5 discusses the experimental results obtained and Section 6 contains the conclusions of the paper and some directions for further research.

2. The Problem of Resource-Constrained Project Scheduling

RCPSP [1] considers a project with a set of activities and a set of available resources. Let J be the number of activities (also called jobs) and $\{a_1, ..., a_J\}$ the set of activities. Jobs have to follow certain precedence constraints which means that a job can not start before all its predecessors are finished. Let us denote by P_j the set of all predecessors of activity a_j and by S_j the set

of all its successors. The precedence constraints are usually represented as an acyclic activity-on-node network. Two additional activities are also normally considered: an initial activity a_0, which must precede all other activities of the project and a final activity a_{J+1} which must be preceded by all activities of the project.

In order to be executed, each activity requires a certain amount of some of the available resources. Let K be the set of existing resources. In this work, we only consider renewable resources - characterized by a constant per-period-availability (R_k, for each $k \in K$). The amount of resource k needed by the activity a_j is denoted by r_{jk}. Furthermore, each activity a_j has a fixed duration or processing time denoted by p_j. It should be noted that the dummy activities (a_0 and a_{J+1}) require no time and no resources.

A schedule assigns a start time to all activities $a_1, a_2..., a_J\}$ with the property that the end time of an activity a_j is the sum of its start time and duration p_j. This makes the project makespan to be equal to the start time of the final dummy activity a_{J+1}. A schedule is feasible if at any time the demand for resource k does not exceed its availability: $\sum_j r_{jk} \leq R_k$.

The goal of RCPSP is to find a schedule of the activities with minimum makespan taking into account the precedence and the resource constraints.

3. The Geometric Collaborative Evolutionary Model

The *GCE* model [2, 3, 5] integrates agent-based behavior into the evolution of the population in order to facilitate the adaptation of individuals to the environment as the search process progresses.

The population has a geometrical structure which allows the definition of a neighborhood notion used in designing different selection strategies (see Figure 1). Initially, all individuals are sorted according to their fitness and distributed over concentric layers starting with the most fit individuals on the most inner layers. The population size is fixed at n^2 (where n is an even number) which leads to a number of $n/2$ layers, each layer i ($i = 0, ..., n/2-1$) having $4(n - 2i - 1)$ individuals. Let us denote the sorted population at iteration t by $P(t) = (x_1, x_2, ..., x_{n^2})$, where x_1 is the fittest and x_{n^2} is the worst individual in the population. The most inner layer contains the first four individuals (x_1, x_2, x_3, x_4). The next layer holds 12 individuals $(x_5, ..., x_{16})$ having the next best fitness values. The most outer layer is labeled by 0 whereas the label of the most inner layer is $n/2 - 1$. The example depicted in Figure 1 uses a population of 8^2 individuals which leads to 28 individuals being placed on layer 0 down to the best 4 individuals placed on layer 3.

Based on this population topology, *local* selection refers to individuals situated on the immediately previous layer (better but still resembling fitness),

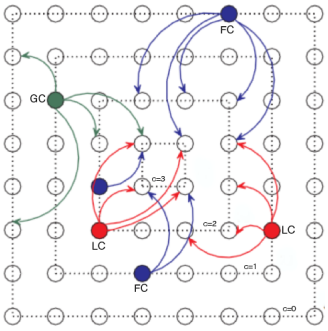


FIGURE 1. GCE Population Topolgy and Society Strategies.

far selection seeks individuals from more distant layers (at least two layers apart) containing more fit individuals while *global* selection considers the entire population. The agent-inspired component of the GCE model refers to the organization of the population in three societies of individuals co-evolving during the search process. Each individual can be viewed as an agent with the objective of optimising its fitness being able to communicate and select a mate for recombination. Individuals belong to one the following three societies: *Local Correlation (LC)*, *Far Correlation (FC)* and *Global Correlation (GC)*.

The GCE model uses an asynchronous search scheme. Individuals from a layer are updated through recombination and are involved in forthcoming recombination processes within the same generation. The individuals from the most inner layer are automatically copied in the next population (as an elitist strategy). Each individual from the population has the chance of being improved by getting involved in a recombination process.

Recombination in the GCE model is guided by the following strategies corresponding to each of the three agent societies (see Figure 1):

- LC individuals from a layer c address mating invitations to individuals from layer $(c+1)$, where $c = 0, ..., n/2-2$.
- FC individuals from a layer c address mating invitations to individuals from layer $(c+i)$, where $c = 0, ..., n/2-3$ and $i \geq 2$ is randomly selected using a uniform distribution. FC individuals from layer $(n/2-2)$ invite individuals from layer $(n/2-1)$.
- GC individuals from a layer c may address mating invitations to individuals from any layer except layer c.

Furthermore, each individual invited to be a mate can accept or decline the proposal according to its own strategy. Normally, individuals from *LC* and *FC* societies accept individuals from the same society or from the *GC* society as mates. Individuals from *GC* society accept any other individual as

mate. Offspring are assigned to a certain society according to a dominance concept. If LC is the dominant agent society then any combination of a GC individual with a LC individual results in an offspring belonging to LC.

A probability of one society dominating another is used to modulate the interactions between individuals belonging to different societies. Let p be the probability of LC (and FC) dominating GC. The dominance probability p may be viewed as the (probabilistic) membership degree of an offspring to the society LC (FC) when one of the parents is a GC individual. Several assignment schemes for p are analysed in [5] focusing on the dynamics given by the co-existence of the three societies of individuals.

For each mating pair (x, y) the offspring z obtained after recombination is mutated obtaining $mut(z)$. The best between z and $mut(z)$ is compared to the first parent x and replaces x if it has a better quality. The elitist scheme that allows only better individuals to replace the first parents is mitigated by the fact that all individuals from the population are involved in recombination.

The importance of the agent-inspired component of GCE has been investigated in [3] and the results support the hypothesis that an adaptive behavior of individuals within an evolving population benefits the search process. This adaptive behavior is triggered in the GCE model by the interactions between individuals belonging to societies with different strategies for selection and recombination.

4. Evolutionary Algorithm for RCPSP

The evolutionary approach to RCPSP developed in the current paper is an instantiation of the GCE model. Therefore, the evolutionary algorithm for RCPSP implements all GCE principles regarding the population topology, the asynchronous search scheme and the LC, FC, GC societies of individuals. This section focuses on specifying the individual representation, the crossover scheme, the mutation operator and the fitness function used in GCE specifically for RCPSP.

The first thing to consider in an evolutionary algorithm for RCPSP is how a solution of the problem is to be encoded in a chromosome. Several different codifications for RCPSP have been proposed in the literature, among which: activity list representation, random key representation, priority rule representation, shift vector representation and schedule scheme representation. This work focuses on the activity list representation: a solution of the problem is encoded as a list of the activities which represent their execution order. If an activity a_2 appears after another activity a_1 in the activity list, it means that the start time of activity a_2 is higher or equal to the start time of activity a_1: $T(a_1) \leq T(a_2)$. The list of activities must be precedence feasible i.e. each

activity must have a higher index than any of its predecessors. An activity list is translated into a schedule by assigning the smallest possible start time to each activity [10]. This means that no activity can be left shifted without violating the constraints.

The Uniform Crossover [4] operator is used for generating offspring in the recombination process. The mutation operator used in the GCE approach to RCPSP swaps two genes of a chromosome provided that the obtained permutation still represents a feasible solution of the problem [4]. Mutation is important in the evolutionary search process as it can lead to new individuals that recombination is not able to obtain.

Finally, the fitness of an individual is defined as the makespan of the corresponding schedule which should be minimized.

5. Computational Experiments and Results

The GCE algorithm is applied to several RCPSP instances and the results are directly compared to a standard evolutionary algorithm (SEA). Both evolutionary algorithms use a population size of 100, mutation rate of 0.05 and 100 generations of evolution. The initial population is randomly generated in such a way that only feasible individuals are obtained. This is done in the following way: (i) the first chromosome gene is randomly selected from the entire activities list, and (ii) next genes are randomly generated from the remaining list of activities as long as all the predecessors of the chosen gene already exist in the chromosome configuration obtained at that point.

In the standard evolutionary approach, roulette selection [4] is used for choosing which individuals should enter the mating pool. The best individual obtained in one generation will always replace one randomly generated individual from the next generation. This mechanism ensures that the best individual obtained in the last generation is actually the best individual obtained in all generations of the algorithm. The GCE model automatically has access to the best individual each generation due to the specific population geometry.

For testing the performance of the proposed model, several ProGen project instances with 60 and 120 activities have been considered [9]. The results obtained after 20 runs of the algorithms for each instance, presented in Table 1 and Table 2, are compaired using the paired t-test with a 95% confidence interval. For the p-values smaller than 0.05 we can conclude that the mean values obtained when using GCE are significantly smaller than those obtained when using SEA. This situation appears for 7 out of the 10 considered project instances with 60 activities and for all 10 considered project instances with 120 activities, even if their complexity is more significant. These test results

indicate the acceleration of the search process when using the proposed GCE model.

TABLE 1. Best and average makespan obtained after 10 runs for each instance with 60 activities

	GCE		SEA		T-test
	Best	Average	Best	Average	p-value
J601_1	81	83.6	85	88	0.007454
J601_2	81	83.6	82	86.9	0.027292
J601_3	73	75.5	75	78.4	0.004937
J601_4	93	95.4	93	98.3	0.045599
J601_5	81	83.5	81	83.2	0.576312
J601_6	66	68.4	71	73.8	0.000067
J601_7	77	80.1	77	82.6	0.136332
J601_8	87	90	89	93.9	0.008539
J601_9	93	97.3	94	97.7	0.583877
J601_10	85	85.8	91	93.4	0.000002

TABLE 2. Best and average makespan obtained after 10 runs for each instance with 120 activities

	GCE		SEA		T-test
	Best	Average	Best	Average	p-value
J1201_1	140	146.4	154	161.2	0.000296
J1201_2	141	147	150	157.7	0.001782
J1201_3	151	154	158	164.2	0.000560
J1201_4	123	128.7	135	139.1	0.000513
J1201_5	148	153.7	155	166.2	0.000265
J1201_6	105	110.8	116	123.1	0.000011
J1201_7	137	146.5	151	158.4	0.000027
J1201_8	140	150.1	156	160.9	0.000183
J1201_9	133	151.7	160	168.5	0.000718
J1201_10	146	152.2	160	164.7	0.000612

6. Conclusions and Future Work

A geometric collaborative evolutionary model is applied for solving the NP-hard optimization problem of resource-constrained project scheduling. The model is based on the geometric topology and the organization of individuals

in different societies with different recombination strategies. The communication between individuals is facilitated by the agent inspired behaviour. Numerical experiments performed on ProGen project instances with 60 and 120 activities emphasize a good performance of the geometric collaborative model. As further work, comparisons with other evolutionary methods for RCPSP will be performed.

7. Acknowledgments

This research is supported by Grant PN II TE 320, Emergence, auto-organization and evolution: New computational models in the study of complex systems, funded by CNCS Romania.

References

[1] J. Blazewicz, J. Lenstra, A. Rinnooy Kan, *Scheduling subject to resource constraints: Classification and complexity.* Discrete Applied Mathematics 5 (1983), 11-24.

[2] C. Chira, A. Gog, D. Dumitrescu, *Exploring Population Geometry and Multi-Agent Systems: A New Approach to Developing Evolutionary Techniques*, GECCO (Companion), 1953-1960, 2008.

[3] C. Chira, A. Gog, D. Dumitrescu, *Asynchronous Collaborative Search using Adaptive Coevolving Subpopulations*, ECOMASS Workshop, Genetic and Evolutionary Computation Conference GECCO'09, F. Rothlauf (Ed), GECCO Companion, ACM, 2575-2582, 2009.

[4] Eiben, A.E., Smith, J.E., *Introduction to Evolutionary Computing*, Springer-Verlag, 2003.

[5] A. Gog, C. Chira, D. Dumitrescu, *Asynchronous Evolutionary Search: Multi-Population Collaboration and Complex Dynamics*, Congress on Evolutionary Computation (CEC 2009), 240-246, 2009.

[6] A. Gog, C. Chira, Recombination operators in permutation-based evolutionary algorithms for the travelling salesman problem, Chapter 10 in Logistics Management and Optimization through Hybrid Artificial Intelligence Systems, IGI Global, (2012) 268-285.

[7] S. Hartmann, *A competitive genetic algorithm for resource-constrained project scheduling.* Tech. Rep. 451, Manuskripte aus den Instituten fur Betriebswirtschaftslehre der Universitat Kiel, (1997).

[8] J.-L. Kim, R.D. Ellis, Jr, *Permutation-Based Elitist Genetic Algorithm for Optimization of Large-Sized Resource-Constrained Project Scheduling*, J. Constr. Eng. Manage. 134, 904 (2008).

[9] R. Kolisch, C. Schwindt, A. Sprecher, *Benchmark instances for project scheduling problems*; Kluwer; Weglarz, J. (Hrsg.): Handbook on recent advances in project scheduling, (1999) 197-212.

[10] R. Kolisch, *Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation*, European Journal of Operational Research, 90, (1996) 320333.

[11] Z. Li, J. Qin, *A new hybrid genetic algorithm and its application in the RCPSP*, Proc. SPIE 8349, 83490Z (2011).

[12] Y. H. Ren, D.C. Kong, W.L. Peng, *A Genetic Algorithm Based Solution with Schedule Mode for RCPSP*, Advanced Materials Research, vol 268-270, (2011), 1802-1805.
[13] H. Zhang, *A Genetic Algorithm for Solving RCPSP*, Computer Science and Computational Technology, 2008. ISCSCT '08. International Symposium on, (2008), 246 249.

Department of Computer Science, Babes-Bolyai University, Kogalniceanu 1, 400084 Cluj-Napoca, Romania
E-mail address: {anca,cchira}@cs.ubbcluj.ro