International Journal of Computers, Communications & Control
Vol. I (2006), No. 1, pp. 33-39

Symbolic Computations based on Grid Services

Dana Petcu, Cosmin Bonchiş, Cornel Izbaşa

Abstract: The widespread adoption of the current Grid technologies is still impeded by a number of problems, one of which is difficulty of developing and implementing Grid-enabled applications. In another dimension, symbolic computation, aiming to automatize the steps of mathematical problem solving, has become in the last years a basis for advanced applications in many areas of computer science.
In this context we have recently analyzed and developed grid-extensions of known tools for symbolic computations. We further present in this paper a case study of a Web service-based Grid application for symbolic computations.
Keywords: Grid computing, Web services, Mathematical software

1 Introduction

It is widely recognized that Grid computing is THE computer buzzword of the decade. Many of the greatest challenges for software systems lie in how to enable automated software components deployed by different organizations to dynamically discover one another, communicate and coordinate their actions and form sound, robust and effective compound applications and services. The Grid promises to solve these problems. The Grid underlying problems are multi-disciplinary and cover a wide range of issues - from service discovery, management, robust management of dependencies and system-system communication to security, legal or ethical frameworks, methodologies, verification, testing, and finally deployment of infrastructure in a shared and accessible environment. After the forerunner first-generation Grid systems, the second-generation proposed the vision of computing resources to be shared like content on the Web over the Internet. The associated layered architecture abstracts fundamental system components, their purpose, and their interaction with each other. Moreover the current, third-generation Grid is aligned with Web services. Comparing to the conventional distributed computing environments, the usual Grid environment focuses on the user: users working on their home machines see the illusion of a single computer, with access to all kinds of data and physical resources. Moreover, the specific machines that are used to execute an application are chosen from the user's point of view, maximizing the performance of that application, regardless of the effect on the system as a whole.

The two basic approaches to computational solution of mathematical problems are numerical and symbolic. For a long time, the numerical approach had an advantage of being capable of solving a substantially larger set of problems. Developments in symbolic computing are lagging relative to numerical computing, mainly due to the inadequacy of available computational resources: most importantly computer memory, but also processor power. Continuous growth in the capabilities of computer hardware led to an increasing interest in symbolic calculations. A transition from numerical modeling to analytical modeling can be observed today in various spheres of science and technology; a motivation consists in the desire to construct more accurate and faithful simulations. Currently software tools for symbolic computations, known as Computer Algebra Systems (CAS), allow users to study computational problems on the basis of their mathematical formulations and to focus on the problems themselves instead of spending time transforming the problems into forms that are numerically solvable.

There are three ways in which a CAS can interact with the Grid, that are detailed in the next section: the CAS uses a Grid service to improve its own services, the CAS uses the Grid infrastructure to improve its response time, or the CAS is presented to a client application as a Grid service. We have studied the first cases in [11] respectively [10]. In this paper we look to the third approach.

The paper is organized as follows. Section 2 presents a state-of-the-art in the field. Section 3 underlines the benefits of using Grid services for symbolic computations. In the context of the current Grid and CAS technologies, a case study is presented in Section 4. Further work directions are identified in Section 5.

2 The Grid and the CAS

2.1 Approaches for CAS-Grid interaction

A Grid-extension of a CAS system can tackle one or more of the following approaches:

Ability to accept services from the Grid: the CAS must be capable to augment its facilities with external modules, in particular it should be able to explore computational Grid facilities, to connect to a specific Grid service, to use it, and to translate its results for the CAS interface. This approach is taken into consideration by NetSolve/GridSolve [1], Geodise [4], MathGridLink [14], our Maple2g [9].

Ability to communicate and cooperate over the Grid: several kernels of CASs must be able cooperate within a Grid when solving problems; in order to have the same CAS on different computational nodes a Grid-version must be available; in the case of different CASs, appropriate interfaces between them must be developed and implemented or a common language for inter-communication must be adopted. This approach is taken into account by Maple2g [10].

Being a source of Grid or Web services: the CAS or some of its facilities must be reachable as grid or web services and allowed to be activated by remote users under appropriate security and licensing conditions; furthermore, deployment of the services must be done in an easy way from the inside of the CAS. This approach is taken into account by the several projects described bellow.

More details about the first two approaches can be found in [12]. Details about the third approach follows.

2.2 Web service-based CAS extensions

In number theory there exist a number of successful Internet projects [6] aiming, among others, at finding large prime numbers, factoring large numbers, computing digits of π, finding collisions on known encryption algorithms etc. A CAS web-wrapper component that can be used by multiple systems was reported in [15]. Another online system, OGB (for Gröbner basis computations), has been recently deployed [5].

MapleNet [7] offers a software platform to enhance mathematics and related courses over the Web. The client machine must be able to run Java applets. A publisher machine is responsible for creating and editing the content of the Web pages and, when complete, uploading them to the server. The server is the machine to which clients will connect to access Web pages and the applets associated with them. The server also respond to publishing requests from the publisher machine for the transfer of content between the publisher and the server. It manages concurrent Maple instances as required to serve client connections for mathematical computation and display services. The server can also provide some additional services including user authentication, logging information, and database access.

WebMathematica [16] offers access to Mathematica applications through a Web browser or other Web clients. Mathematica can be seen as a development environment for webMathematica sites. Standard Java technologies are used: Java Servlet and JavaServer Pages. WebMathematica allows a site to deliver HTML pages that are enhanced by the addition of Mathematica commands. When a request is made for one of these pages, the Mathematica commands are evaluated and the computed result is inserted into the page. Input can come from HTML forms, applets, JavaScript, and Web-enabled applications. It is also possible to send data files to a server for processing. Output can use different formats such as HTML, images, Mathematica notebooks, MathML, SVG, XML, PostScript, Pdf.

The Monet project [8], funded by the European Commission, was a two-year (April 2002-March 2004) investigation into mathematical Web services aiming to demonstrate the applicability of the Semantic Web to the world of mathematical software. The principal objective was to develop a framework for the description and provision of Web-based mathematical services. The key to such a framework is the ability to discover services dynamically based on published descriptions which describe both their mathematical and non-mathematical attributes. Such discovery and subsequent interaction are mediated by software agents capable of recognizing the criteria which should determine how particular kinds of problems are solved, and extracting them from the user's problem description. A symbolic solver wrapper tool architecture was designed to provide an environment that encapsulates CASs and expose their functionalities through symbolic services deployed. All symbolic services are running as independent Web services, each reachable at its own unique URL, all of them are enclosed within the symbolic server and they are managed by the wrapper tool symbolic solver environment. Each symbolic service is assigned to several instances, such as a service core Java class, a source code implementing the service with a mathematical solving software (a CAS), and a MSDL file. The principal information about each service is provided by the service configuration file that contains tree parts: service's MSDL, service interface to mathematical solving system and the actual service implementation. The technologies used for symbolic solver services implementation are Java, Axis, Tomcat, SOAP, WSDL, JSP, MSDL. Maple was chosen as an example of the solving engine for the first implementation and Axiom was used to validate the architecture and to demonstrate abilities to adopt different solving engine without performing major changes.

2.3 Grid service-based CAS extensions

There exist a number of grid-oriented projects that involve CASs.

The project Grid Enabled Numerical and Symbolic Services [3], GENSS (March 2004-February 2006), in the frame of UK e-Science programme, addressed the combination of Grid computing and mathematical Web services, their extension to deliver mathematical problem analysis, the code and the resources to compute the answers, using a common open agent-based framework. The main research focus lied on matchmaking techniques for advertisement and discovery of mathematical services. The project involved the design and implementation of an ontology for symbolic mathematical problems and used to support service specification and registration of services. The ontology has been extended based on work undertaken in Monet [8].

The Grid Execution Management for Legacy Code Architecture, GEMLCA [2] is a recent solution to deploy existing legacy code applications written in any programming language as a Grid service without modifying or even requiring access to the legacy code (source or binary). The access point for a client to the GEMLCA architecture is the front-end layer composed of a number of Grid services offering interfaces in order to deploy, query, submit, check the status of, and get the results back from computational jobs. The front-end layer is described in WSDL and can be used by any Grid service client to bind and make use of its functionality through SOAP. In order to access a legacy code program, the user executes the GEMLCA grid service client that creates a legacy code instance with the help of a legacy code factory. Following this, the system submits the job to the compute server through Globus Toolkit version 3 using a particular job manager. A specific XML format, LCID (Legacy Code Interface Description File) is necessary to be used.

3 Benefits of Using Symbolic-computing Services based on Globus-WSRF

The recent version 4 of Globus Toolkit, de-facto standard for Grid technologies, is based on standard Web Services technologies such as SOAP and WSDL. It is written according the WSRF specification (Web Services Resource Framework). The basic requirements addressed by WSRF is the ability to create, address, inspect, discover, and manage statefull resources. Grid services extends Web services (usually stateless services) by providing these extra functionalities.

WSRF approach is more flexible than the previous ones implemented in Globus Toolkit (e.g. OGSI implementation) allowing many-to-many mappings between Web services (the message processor) and any associated statefull resource (the statefull service instance).

3.1 Statefull services

The WSRF approach simplifies the development of Grid-service wrappers for CASs. The CAS can take now the role of the statefull service instance.

If we go back to the case of the Web-wrapper of a CAS, we can identify several problems solved by the Grid environment. Successive related requests to the service hiding the CAS will need the maintenance of the service instance in a command waiting cycle, without releasing the connections. When the connection is interrupted, but the client come back to the system it must start as a any new incomer.

If a statefull service is used, the latest state of the CAS can be registered. Despite the fact that the connection was closed, the client can come back and resume the computation at any time before the service instance destruction.

3.2 Service instances on remote Grid nodes

Using a appropriate scheduler the CAS service instance can run on a different Grid hardware resource than the one where the container for Web-based Grid services resides, primarily addressed by the service client. This approach solves the problem of the server overload of a classic client-server architecture.

3.3 Service discovery

The user is confronted with thousands of packages are available to perform all kinds of mathematical computations. A standard way to categorize, explore, discover, invoke and compose them is needed. Grid computing has

awake high expectations for its potential as a discovery accelerator. Grid WSRF-based services are described in WSDL standard format that can be easily inspected by any potential client.

3.4 Security standards

Globus's GSI offers two message-level protection schemes, and one transport-level scheme [13]. GSI Secure Message scheme that provides message-level security can be used in the case on proprietary CAS usage.

Globus Toolkit implements three authentication methods: X.509 certificates, username-password, and anonymous authentication. The first two authentication methods are recommended in the case of a proprietary CAS usage.

GSI supports also authorization in both the server-side and the client-side. The server has six possible authorization modes: none, self, gridmap, identity-authorization, host authorization, SAML callout authorization. Depending on the authorization mode that if will be chosen, the server will decide if it accepts or declines an incoming request. Identity-authorization or SAML callout authorization are recommended in the case of a proprietary CAS usage.

4 A case study: Grid-based services using Maple

We proceed with a practical example of how a CAS can be made available as Grid-WSRF service and for this purpose Maple became our CAS of choice. The main reason is that, despite its robustness and ease of use, we were not able to locate efforts to link Maple with the Grid, accept ours, namely Maple2g. Furthermore, it is well known that Maple excels other CASs in solving selected classes of problems e.g. systems of nonlinear equations or inequalities. Finally, Maple has already a build-in socket library for communicating over the Internet, and a library for parsing XML. These capabilities match very well with our goal as they suffice to make Maple a client for an Grid computational service.

Maple2g (Maple-to-Grid) was build recently as a grid-wrapper for Maple. Maple2g consists of two parts a CAS-dependent and a Grid-dependent one. Therefore, any change in the CAS or in the Grid will be reflected only in one part of the proposed system. Furthermore, the CAS-dependent part is relatively simple and easy to be ported to support another CAS or legacy software.

Maple2g covers all three approaches described in Section 2.1. We describe here the newest component, the one that presents Maple as Grid service. It is based on the WSRF implementation from Globus Toolkit 4.

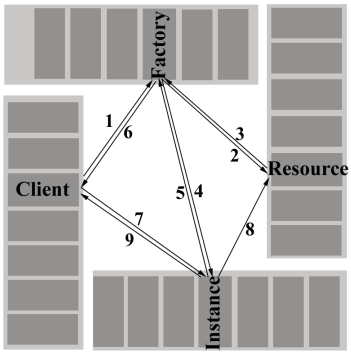

Figure 1: The four elements of the application, the client, the factory, the instance, and the resource. Their interaction in time

At the server side, the application has two main components: the factory service, the instance service. A resource is created by the factory service and used by the instance service. The interaction pushed by the client follows the steps (Figure 1):

1. the client request the service instance and resource creation;
2. the factory service creates the resource;
3. the key of the resource is identified by the factory;
4. the factory creates the service instance;

5. the service reference is returned to the factory;
6. the endpointreference (instance and resource identifiers) is send to the client;
7. the client contacts the service instance and request an operation involving the resource;
8. the service instance serves the request using the resource;
9. the service instance sends the result to the client.

The client can reiterate the steps (7-9) using the endpointreference.

The application components were written in Java. The factory service is activated once the Globus service container is activated on the resource were the service package was deployed. The instance service launches Maple as a concurrent thread on the same hardware resources on which it runs.

The communication between the Java-based instance service and Maple is done via sockets. At start Maple thread reads a temporary file that specify the socket channel that should be open and then the cycle in which it accepts any string coming on that communication channel, evaluate it as Maple command, and sends the result via the socket connection to the instance service.

For example, after the MapleFactoryService activation, the MapleClientPerform can send Maple commands to the instance service in form of strings:

```
$ MapleClientCreate http://194.102.62.15:8080/wsrf/services/MapleFactoryService >ref

$ MapleClientPerform ref "ifactor(12345678905678909876509876543210 0);"
(2)^2*(5)^2*(7)*(43)*(113)*(331)*(683)*(12067378391)*(65837)*(202087)

$ MapleClientPerform ref "P:=12*x^6+84*x^4-54*x^5-270*x^3+168*x^2-216*x+96:"

$ MapleClientPerform ref "solve(P);"
1/2, 4, I, -I, 2*I, -2*I
```

We have build also a Web interface that is depicted by Figure 2. We used AJAX (Asynchronous JavaScript And XML) for sending and receiving dates to the server. After the user authentification, the interface can be use in few simple step:

1. start the Globus container on the server (Figure 2.a);
2. choose the proper service (MathFactoryService) from the list of available services (Figure 2.b);
3. start the MathFactoryService which will launch the Maple instance (Figure 2.c);
4. fill and execute a Maple command (Figure 2.d and e).
5. the Maple instance executes the command and returns an MathML document as answer. The MathML document will be transform to a SVG file which is send back to the interface (Figure 2.f).

5 Conclusions and Further Work

At this stage Maple2g exists as a demonstrator system with some of the functionalities described above. In the near future it will be further developed to include facilities existing in other systems, in order for it to become more robust.

Currently if the access to Maple service is granted, any Maple commands can be used. Restricted access to a subset of commands (e.g. no access to shell commands revealing the host characteristics or establishing socket connections) should by implemented. Specialized services based on Maple should be developed and deployed as Grid services.

Experiments on the wide-area Grids will help guide further development of the system. Deployment of Grid services based on other CASs than Maple using the same codes must be take also into consideration.

Acknowledgment

This work was partially supported by the European project SCIEnce (FP6-2004-Infrastructure-5-026123) and the Romanian project CompGrid (CNCSIS-2004-949).

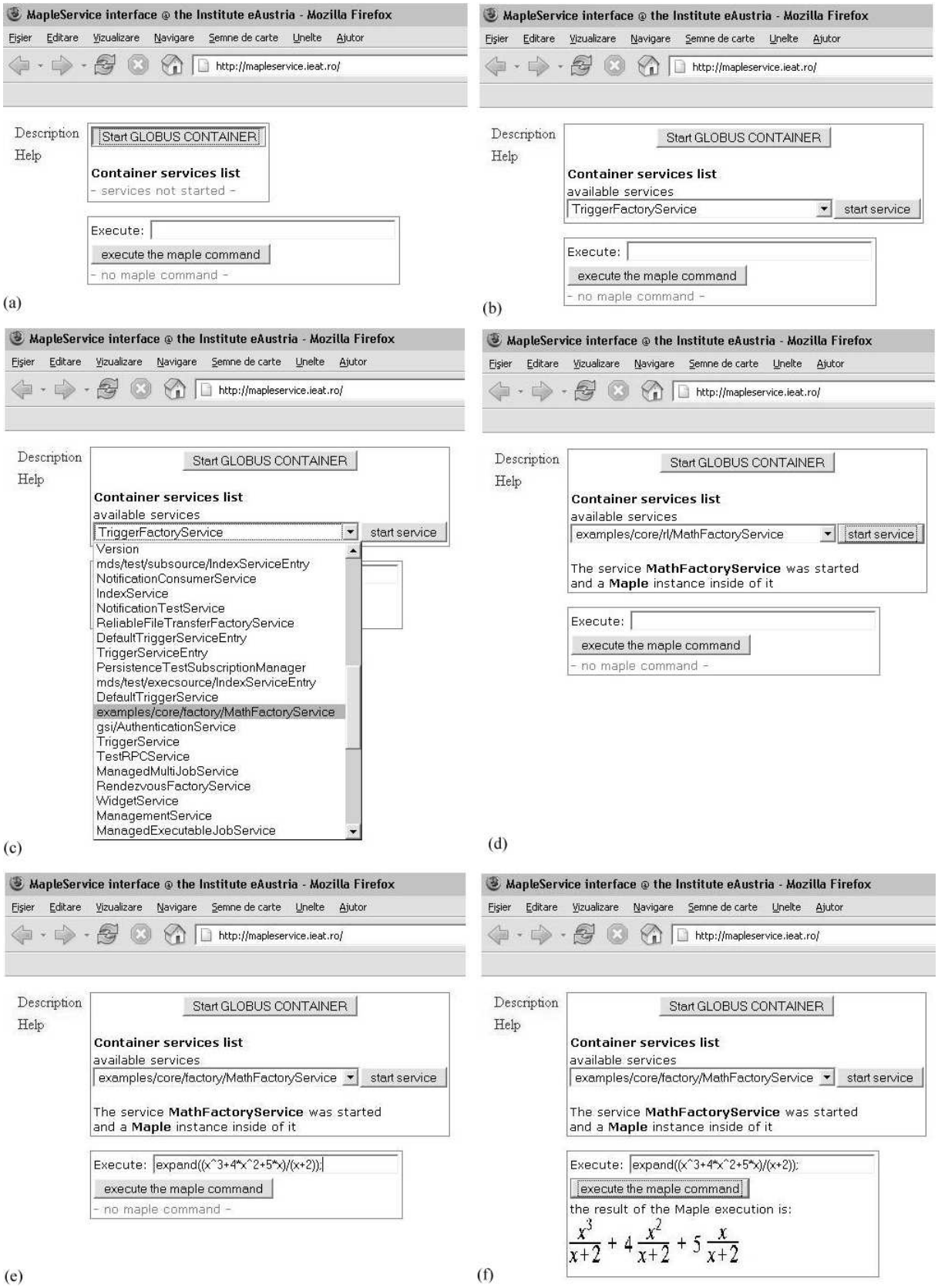

Figure 2: Using a Web interface to access symbolic computing services

References

[1] S. Agrawal, J. Dongarra, K. Seymour, S. Vadhiyar, “NetSolve: Past, Present, and Future – A Look at a Grid Enabled Server", in *Making the Global Infrastructure a Reality*, Wiley, Berman F. (Ed.), pp. 613-622, 2003.

[2] T. Delaitre, A. Goyeneche, P. Kacsuk, T. Kiss, G.Z. Terstyanszky, S.C. Winter, “GEMLCA: Grid Execution Management for Legacy Code Architecture Design”, in *Procs. of the 30th EUROMICRO conference*, Special Session on Advances in Web Computing, August 2004, Rennes, France, pp. 305-315, 2004.

[3] Genss, Available on-line at: http://genss.cs.bath.ac.uk/index.htm.

[4] Geodise, Available on-line at: http://www.geodise.org/.

[5] M. Gettrick, “OGB: Online Gröbner Bases”, http://grobner.it.nuigalway.ie, 2004.

[6] Internet-based Distributed Computing, Available on-line at: http://www.aspenleaf.com/ distributed/ap-math.html.

[7] MapleNet, Available on-line at: http://www.maplesoft.com/maplenet/.

[8] Monet, Available on-line at: http://monet.nag.co.uk.

[9] D. Petcu, D. Dubu, “An Extension of Maple for Grid and Cluster Computing”, *Procs. ICCC 2004*, I. Dziţac, T. Maghiar, C.Popescu (eds.), Oradea, Băile Felix SPA, May 27-29 2004, Ed. Metropolis, pp. 355-360, 2004.

[10] D.Petcu, D.Dubu, M.Paprzycki, “Grid-based Parallel Maple”, *LNCS* 3241, Procs. PVMMPI 2004, Budapest, Hungary, September 19-22, 2004, D. Kranzmüller, P. Kacsuk, J. Dongarra (eds.), Springer, pp. 215-223, 2004.

[11] D.Petcu, M.Paprycki, D.Dubu, “Design and Implementation of a Grid Extension of Maple”, *Scientific Programming*, vol. 13, no. 2, IOS Press, pp. 137-149, 2005.

[12] D. Petcu, D. Ţepeneu, M. Paprzycki, T. Ida, “Symbolic Computations on Grids”, Chapter 27, *Engineering the Grid: Status and Perspective*, B. di Martino, J. Dongarra, A. Hoisie, L. Yang, and H. Zima (eds.), 2006.

[13] B. Sotomayor, L. Childers, *Globus Toolkit 4 : Programming Java Services*, Morgan Kaufmann, 2005.

[14] D. Ţepeneu, T. Ida, “MathGridLink - Connecting Mathematica to the Grid”, in *Procs. IMS’04*, Banff, Alberta, Canada, 2004.

[15] A. Weber, W. Küchlin, B. Eggers, V. Simonis, “Parallel computer algebra software as a web component”, Available on-line at: http://www.cs.ucsb.edu/conferences/ java98/papers/algebra.pdf, 1998.

[16] webMathematica, Available on-line at: http://www.wolfram.com/products/webmathematica/.

Dana Petcu
Institute e-Austria Timişoara and
Western University of Timişoara
Computer Science Department
Address: B-dul Vasile Pârvan 4, 300223
Timişoara, Romania
E-mail: petcu@info.uvt.ro

Cosmin Bonchiş
Institute e-Austria Timişoara, and
Vasile Goldiş University of Arad
Computer Science Department

Cornel Izbaşa
Institute e-Austria Timişoara, and
Western University of Timişoara
Computer Science Department