Doktorandský den ’99

Ústav informatiky
Akademie věd České republiky

Praha, 10. – 11. listopad 1999

Obsah

Doktorandský den ÚI AV '99 – program

— 10. listopad 1999 —

Časový rozvrh přednášek		
Dušan Bálek	Connectors in Component-Based Systems	$8^{00} - 8^{35}$
Radek Pospíšil	On Enhancement of Enterprise JavaBeans Components	$8^{35} - 9^{10}$
	přestávka 10 minut	
Petr Hanzlíček	Decision support in medicine using electronic patient records	$9^{20} - 9^{55}$
Jan Klaschka	Transformace a regrese	$9^{55} - 10^{30}$
	přestávka 10 minut	
Arnošt Štědrý	The joy of Joyce	$10^{40} - 11^{15}$

— 11. listopad 1999 —

Časový rozvrh přednášek		
Ctirad Mátonoha	Trust region methods for large-scale optimization problems	$8^{00} - 8^{35}$
Zdeněk Kestřánek	H-verze kontaktního problému v termopružnosti	$8^{35} - 9^{10}$
	přestávka 10 minut	
Přemysl Žák	Buněčná síť - model počítače s proměnlivou architekturou	$9^{20} - 9^{55}$
Pavel Krušina	Hybrid Methods of Artificial Intelligence	$9^{55} - 10^{30}$

Connectors in Component-Based Systems

doktorand:
MGR. DUŠAN BÁLEK
Malostranské nám. 25, Praha 1, 118 00
balek@nenya.ms.mff.cuni.cz

školitel:
DOC. ING. FRANTIŠEK PLÁŠIL, CSC.
Malostranské nám. 25, Praha 1, 118 00
plasil@nenya.ms.mff.cuni.cz

obor studia:
I2 - Softwarové systémy

Abstrakt

Nowadays, to allow for rapid software evolution, more and more software developers start to construct their products from reusable software components. The architecture of a system is described as a collection of components along with the interactions among these components. Even though the system's main building blocks are components, the properties of the system depend strongly on the character of the component interconnections. This fact gave birth to a connector as an abstraction that provides software developers with a way to capture the nature of connections among components.

In this paper, the authors aim at analyzing the variety of possible roles that connectors can play in component-based systems, and proposing a connector architecture best fitting most of the identified connector roles. This is illustrated on a case study of connectors designed for the SOFA/DCUP component model.

1. Introduction

A few years ago, the trend to construct software systems as a collection of cooperating reusable components appeared and has become widely accepted since. Apart from several academic research projects dealing with components [4, 6, 17, 5, 2, 1, 16], several industrial systems are now also on the market [22, 23, 13, 14, 24].

As for *components*, there is a broad agreement on grasping them as reusable black/grey-box entities with a well-defined interface and specified behavior. Usually, a component can have multiple interfaces; some of them to provide services to the component's clients, other to require services from the environment. Components can be nested to form hierarchies; a higher-level component can be composed from several mutually interconnected, cooperating subcomponents. To describe a component's interface and its architecture, many so-called *architecture description languages (ADLs)* [4, 17, 2, 1, 11] have been designed.

Connectors are architectural elements representing component interconnections. Even though the notion of connectors originates in the earliest papers on software architectures [19], no general consensus on whether connectors are really necessary has been reached until now. There has been a big struggle among researchers; protagonists of this idea argue for connectors saying that communication mechanisms have to be separated from components to increase components reusability (the same component can be used in a variety of environments with different communication primitives); their opponents challenge this by stating that there is a little difference between connectors and components and therefore there is no need for distinct abstractions (communication mechanisms can be separated from "business" components to form dedicated "communication" components).

1.1. The goal of the paper

Originally, the SOFA/DCUP component model [8] did not include explicit connectors. Interconnections among components were primitive bindings of exported interfaces; any non-trivial interconnection semantics was supposed to be modeled via special "communication" components. But, naturally, it became obvious that a communication component differs from a business one in several significant aspects. (1) A "business" component is usually deployed on a particular node in the network, while the "communication" component usually spans several network locations depending on the deployment of the "business" components it connects. (2) "Business" components are autonomous elements encapsulating functionality, while "communication" components exist only to serve the communication needs of others [18]. (3) "Communication" component types are more generic than the component types of "business" components (e.g. a type of a communication component need to be parameterized by the actual interfaces of the components to be connected). These facts led us to the decision to adopt connectors to the SOFA/DCUP component model to replace the non-explicit "communication" components.

The main goal of this paper is to identify and analyze possible roles of connectors in component-based systems, and, based on the analysis, to propose a connector architecture fitting well the roles identified.

The paper has the following outline - a brief overview of the existing approaches to specifying component interconnections is presented in Section 2. Section 3 summarizes the identified variety of possible roles of connectors in component-based systems. A conceptual design of connectors is proposed in Section 4. As a proof of the concept, connectors are adopted to the SOFA/DCUP component model in Section 5. Finally, the main achievements and future intentions are summarized in the concluding Section 6.

2. Connectors in Current Component Models

¿From the related work [4, 17, 2, 1, 11], three basic approaches to specifying component interconnections can be identified. According to the widely accepted terminology [20, 3], these are:

- *Implicit connections.* A typical representative of this approach is the Darwin language [4]. The connections among components are specified in terms of direct binding of *requires* to *provides* interfaces. The semantics of a connection is defined by the underlying environment (programming language, communication primitives of the underlying operating system, etc.) and the communicating components should be aware of it (e.g. to communicate, Darwin components directly use ports in the underlying Regis environment).

- *An enumerated set of built-in connectors.* A typical representative of this approach is the UniCon language [17]. When specifying the component interconnections, the developer is provided with a selection of several predefined built-in connector types. Usually, all the built-in connector types are intended to correspond to the usual communication primitives supported by the underlying language or operating system (such as RPC, pipe, etc.). The semantics of a connection is simply defined by the selected connector type.

- *User defined connecors.* A typical representative of this approach is the Wright language [2]. The connections among components are fully specified by system developers. The expressive power of any language that belongs to this group should be rich enough to specify the behavior of such user-defined connectors. For example, Wright uses modified Hoare's CSP notation to specify sophisticated protocols of component interconnections.

3. Possible Roles of Connectors

In this section, understanding a *connector* to be a first class entity representing a connection between (among) two (several) components, we identify a variety of orthogonal roles that a connector can play in a component-based system. Each of these roles covers a different aspect of a connection.

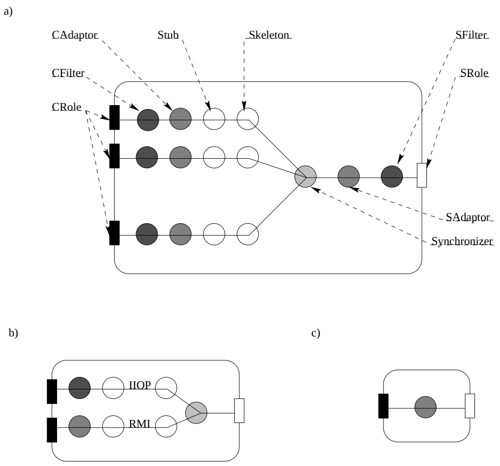
- *Connectors and communication techniques.* The most obvious role of connectors is to specify the communication technique used to implement a particular connection. A connector specifies whether the communication is based on a procedure call, sending events, or, e.g., a data stream. Each of these techniques has its own more detailed characteristics. A procedure call can be local or remote. In the case of remote procedure call, various middleware can be used to implement such a call - e.g. CORBA and Java RMI. A mechanism of sending events can be based for example on an event channel, the publisher-subscriber pattern, or a centralized event queue. Moreover, for a selected communication technology, a connector can specify quality of service, i.e. whether a connection is optimized for performance, reliability, security, saving network resources, etc.
- *Connectors as interface adaptors and data convertors.* While building an application composed from reusable components, a system developer can encounter a need for connecting two (or more) components through interfaces which are incompatible. If the corresponding interfaces are "similar enough", a possible solution is to mediate such a connection via an adaptor converting the calls between these interfaces. A straightforward idea implied by this thought is to conceptually include an adaptor into the connector abstraction. A challenge for a future work is to devise a mechanism for automatic or semi-automatic generation of all necessary interface adaptors and/or data convertors [15].
- *Connectors as access coordinators and synchronizers.* In some component models, the ordering of method calls on component's interfaces is important. The appropriate ordering alternatives are usually specified as a part of the behavioral specification of the component (e.g., interface protocols in SOFA [9], CSP notation in Wright). The synchronization of client calls on component's interfaces could be one of the connectors' tasks.
- *Connectors as communication interceptors.* Connector is a handy abstraction to intercept component communication without letting the participating components be aware of it. An interception can be used for several reasons - to implement filters for the purpose of cryptography and/or data compression, to implement mechanisms for a load monitoring of a particular component in the system, to implement a mechanism supporting debugging (breakpoints), etc.

4. Conceptual Design of Connectors

The proposed connector design (reflecting orthogonality of the connector's roles identified in the Section 3) has been inspired by an idea of open binding presented in [21]. Like a component, *connector*, as an instance of a *connector type*, is an architectural element with well-defined interface, architecture, and behavior. To illustrate the key concepts of the design, *CSProcCall* (client-server procedure call) connector type (Figure 1a) is used.

4.1. Connector interface

A connector interface consist of (possibly) multiple provides/requires interfaces. In contrast to component, connector interfaces are of two kinds: control and conntecting interfaces. A control interface servers for an explicit interaction with the connector (e.g. to perform a control or



Obrázek 1: An example connector architecture: CSProcCall connector type architecture (a), example instances of CSProcCall connector type (b, c)

management operations), while a connecting interface is reflect the "connecting role" of the connector - it is designed to support an interaction among components. Connecting interfaces are typically abstract (generic) - it means that a concrete interface is substituted for an abstract one at the time of the connector's instantiation (*CSProcCall*'s abstract interfaces are CRole$i$ and SRole). For example, instances of the *CSProcCall* connector type can be used to issue a remote call upon all possible server interfaces.

4.2. Connector architecture

A connector architecture describes an internal structure of a connector in terms of mutually connected connector elements. Each of these elements is supposed to be responsible for performing a particular role of the connector. For example, the *CSProcCall* connector type architecture consists of the following elements: CFilter and SFilter (responsible for filtering communication), CAdaptor and SAdaptor (responsible for adapting interfaces), Stub and Skeleton (responsible for performing remote procedure call), and Synchronizer (responsible for clients' synchronization). Most of the connector elements are abstract (generic); concrete elements are substituted for abstract ones at the time of connector instantiation. Some of the substituted concrete elements can be null. Examples of the two possible *CSProcCall*'s instantiations are depicted in Figures 1b, 1c.

Note that connectors can be nested; a connector type can use an instance of another connector type as its internal element.

4.3. Tailoring of connector instances

By tailoring of a connector instance we mean a process of the decision what concrete interfaces and/or elements to substitute for the abstract interfaces and/or abstract elements of the corresponding connector type. The process of tailoring has several phases. The first phase runs during specification a system's architecture. Based on the type of components the particular connector instance connects, its concrete interfaces can be deduced. Also, a need for an interface adaptor and a synchronizer to be employed may become evident. During the system deployment phase, a deployment of a particular connector is derived from the deployment of components connected by the connector. The need for elements performing remote communication arises for a connector spanning several network locations. Some of the elements can be tailored at runtime (e.g. setting up breakpoints).

5. Case Study

As a proof of the concept, connectors conceptually described in Section 4 have been embedded in the SOFA/DCUP component model.

5.1. SOFA/DCUP component model

In SOFA [8], an application is viewed as a hierarchy of software components. In analogy with the classical concept of an object as an instance of a class, a *software component* is an instance of a *component template*. A "template" is referred to as a "component type".

Basically, a template is a pair ¡template frame, template architecture¿. *Template frame* of the template T defines interfaces of the T's instances. These interfaces are specified as sets of services either *provided* or *required*. Simply, template frame provides black-box view on T's instances. To provide gray-box view on T's instances, *template architecture* is used. It describes the internal structure of a concrete T's instance in the terms of its direct subcomponent instances and their mutual interconnections (*ties*). The template architecture can be specified as *primitive* which means that there are no subcomponents and the frame is directly implemented using an underlaying implementation language. The *SOFA CDL* (Component Definition Language) [11] is used to specify a template's frame and its architecture.

The DCUP architecture is a specific architecture of SOFA components which allows for their safe updating at runtime. By *updating* we mean a process of changing component architecture within a frame. For more details on structure of DCUP components and dynamic updates we refer the reader to [8].

5.2. SOFA/DCUP connectors

In analogy with SOFA components, a connector in SOFA is a tailored instance of a *connector template*. A "connector template" can be interpreted as a "connector type". Alike a component template, a connector template is a pair ¡connector template frame, connector template architecture¿.

A *connector frame* defines a connector's interfaces using *provides* and *requires* clauses. The difference to a component frame is that some of interfaces are abstract (see connecting interfaces in Section 4.1). A *connector architecture* specifies a connector internal structure in terms of nested elements and their mutual binding. Some of the internal elements are generic; a concrete elements are tailored at a connector instantiation time.

The DCUP architecture allows for safe connector updates through a connector's *ConnectorUpdateInteface*.

To save developers time otherwise needed to specify a frequently used connector types repeatedly, SOFA/DCUP predefines a set of basic connector types. Predefined connector types with well-defined interfaces and architectures in SOFA/DCUP are supposed to be *CSProcCall* for a client-server communication based on procedure calls, *EventDelivery* for a communication based on sending events, and *DataStream* data stream for a communication based on data flow. The architecture of the *CSProcCall* connector type is depicted in Figure 1 (Section 4). The fragment of its CDL specification looks as follows:

```
...
connector frame CSProcCall {
  provides: abstract CRole[];
  requires: abstract SRole;
};

connector architecture CSProcCall version v1 {
  inst CFilter cf[];
  inst CAdaptor ca[];
  inst Stub st[];
  inst Skeleton skel[];
  inst Synchronizer sync;
  inst SAdaptor sa;
  inst SFilter sf;

  delegate CRole[] to cf[].Iprov;
```

```
  bind cf[].Ireq to ca[].Iprov;
  bind ca[].Ireq to st[].Iprov;
  bind st[].Ireq to skel[].Iprov;
  bind skel[].Ireq to sync.Iprov;
  bind sync.Ireq to sa.Iprov;
  bind sa[].Ireq to sf.Iprov;
  subsume sf.Ireq to SRole;
};
```

For sake of brevity, the CDL specification of the other two predefined connector types is omitted in this paper.

5.3. CDL specification language

To specify component s interfaces, behavior, and architecture, SOFA CDL (ComponentDefinition Language) has been designed. With introducing connectors as first class entities to SOFA/DCUP component model, a need for minor modification of this language has naturally arisen. These modifications include the new keywords *using* and *bind together*. Desired usage of these elements illustrates the fragment of the CDL specification describing the bank dealing system application.

```
...
system BankDealing version v1 {
  inst Receiver rec;
  inst Parser pars;
  inst Supervisor sup;
  inst DataStore ds;
  for i=1 to 4 do
    inst Dealer dealer[I];
  done
  inst DataStream C3;
  inst EventDelivery C4;
  inst CSProcCall C5, C6;
  bind rec.cons to pars.cons using C3;
  bind together pars.l, d[1].up, d[2].up, d[3].up, d[4].up using C4;
  bind together d[1].ap, d[2].ap, d[3].ap, d[4].ap, sup.si using C5;
  bind together d[1].ds, d[2].ds, d[3].ds, d[4].ds, ds.dsi using C6;
};
```

The stream of business news from all of the world's stock exchanges is received from the Reuter's satellite broadcasting and parsed to filter the relevant information. The selected information is send in a form of an event to the bank's dealers to help in their decision whether to buy or sell a stock. Each new deal is inserted to a database. Deals exceeding some limit have to be approved by supervisor. At least three different communication technologies can be found in this scenario: a data flow between the *Receiver* component and the *Parser*, sending events from the *Parser* to bank's *Dealers*, and *Dealer*'s procedure calls on the *DataStore* and *Supervisor* components.

6. Conclusion

In this paper, we have supported the idea of introducing connectors as first class entities in component-based systems. The benefit of this approach is a clear separation of components and their communication mechanisms which increases component reusability. We have relinquished our (original) approach based on introducing communication components (as a special case of bussiness components) because we identified inherently specific properties in communication components deployment, genericity, and autonomy. (As an aside, in our opinion a significant drawback of the Darwin language is the absence of any kind of communication separation).

Furthermore, we have identified the possible roles of connectors in a component-based system, and have provided a conceptual design of connectors reflecting the orthogonality of the roles identified. The developer is free in construction of new (user) connector types with an arbitrary combination of the orthogonal roles of connectors (in this respect, our approach is not so restrictive in defining connector types as the UniConn's approach with a fixed set of predefined connector types). Also retaining SOFA/DCUP connectors in the implementation (advantage over the Wright language) has some benefits - connectors can help in, e.g., load balancing, debugging, and dynamic system reconfiguration (through the mechanism of DCUP connector's updating).

Our future work will be focused on finding techniques allowing for an automatic (or at least semi-automatic) generation of connector elements, including interface adaptors, stubs and skeletons for remote communication, etc. We believe that this could be done by defining a mapping of every abstract connector element to the programming language used to implement concrete connector elements. Based on the mapping defined, a concrete element will be generated each time a concrete value of an abstract element's parameters is set. Another issue we intend to target is to design some kind of deployment descriptors. This work could be inspired by the CORBA Component Model's deployment descriptors which are based on the XML notation.

References

[1] P. Oreizy: "Issues in the Runtime Modification of Software Architectures", *Tech. rep. UCI-ICS-TR-96-35*, University of California, Irvine, 1996

[2] R. J. Allen: "A Formal Approach to Software Architecture", *Ph.D. Thesis*, School of Computer Science, Carnegie Mellon University, Pittsburgh, 1997

[3] N. Medvidovic, R. N. Taylor: "A Framework for Classifying and Comparing Architecture Description Languages", *In Proceedings of 6th European Software Engineering Conference / 5th ACM SIGSOFT Symposium on the Foundations of Software Engineering*, 1997

[4] J. Magee, N. Dulay, J. Kramer: "Regis: A ConstructiveDevelopment Environment for Distributed Programs", *Distributed Systems Engineering Journal*, 1(5), 1994

[5] D. C. Luckham, J. J. Kenney, L. M. Augustin, J. Vera, D.Bryan, W. Mann: "Specification and Analysis of System Architecture Using Rapide", *IEEE Transactions on Software Engineering*, 21(4), 1995

[6] J. M. Purtilo: "The Polylith Software Bus", *ACM Transactions on Programming Languages and Systems*, 16(1), 1994

[7] F. Plasil, D. Mikusik: "Inheriting Synchronization Protocols via Sound Enrichment Rules", *In Proceedings of Joint Modular Programming Languages Conference*, Springer LNCS 1204,1997

[8] F. Plasil, D. Balek, R. Janecek: "SOFA/DCUP: Architecture for Component Trading and Dynamic Updating", *Proceedings of the Fourth International Conference on Configurable Distributed Systems*, Annapolis, Maryland, USA, IEEE CS

[9] F. Plasil, M. Besta, S. Visnovsky: "Bounding Component Behavior via Protocols" *Proceedings of the TOOLS USA '99*, 1999

[10] F. Plasil, M. Stal: "An Architectural View of Distributed Objects and Components in CORBA, Java RMI and COM/DCOM" *Software Concepts & Tools*, vol. 19, no. 1, Springer 1998

[11] V. Mencl: "Component Definition Language", *Master Thesis, Dep. of SW Engineering*, Charles University, Prague, 1998.

[12] C. Szyperski: "Component Software, Beyond Object-Oriented Programming" *Addison-Wesley*, 1997

[13] OMG: "CORBA Component Model. Volume 1" , *orbos/99-04-16*, 1999

[14] OMG: "CORBA Component Model, Volume 2" *orbos/99-04-17*, 1999

[15] D. M. Yellin, R. E. Strom: "Interfaces, Protocols, and the Semi-Automatic Construction Of Software Adaptors" *Proceedings of the OOPSLA'94*, 1994

[16] V. Issarny, Ch. Bidan: "Aster: A Framework for Sound Customization of Distributed Runtime Systems". *Proceedings of the Sixteenth International Conference on Distributed Computing Systems*, 1996

[17] M. Shaw, R. DeLine, D. V. Klein, T. L. Ross, D. M. Young, G. Zalesnik: "Abstractions for Software Architecture and Tools to Support Them", *IEEE Transactions on Software Engineering*, 21(4), 1995

[18] M. Jackson: "Software Requirements & Specifications: A Lexicon of Practice, Principles and Prejudices" *ACM Press/Addison-Wesley*, 1995

[19] D. E. Perry, A. L. Wolf: "Foundations for the Study of Software Architecture" *ACM Software Engineering Notes*, vol. 17, no. 4, 1992

[20] J. Bishop, R. Faria: "Connectors in Configuration Programming Languages: are They Necessary?" *Proceedings of the Third International Conference on Configurable Distributed Systems*, 1996

[21] G. S. Blair, G. Coulson, P. Robin, M. Papathomas: " An Architecture for Next Generation Middleware" *Proceedings of Middleware'98*, 1998

[22] Sun Microsystems: "JavaBeans 1.0 Specification" http://splash.javasoft.com/beans/spec.html

[23] JavaSoft: "Enterprise JavaBeans 1.1 Specification" http://java.sun.com/products/ejb/docs.html

[24] D. Rogerson: "Inside COM" *Microsoft Press*, 1997

On Enhancement of Enterprise JavaBeans Components

doktorand:
MGR. RADEK POSPÍŠIL
KSI MFF UK, Malostranské náměstí 25, 110 00 Prague 1
rpos@uivt.cas.cz,pospisil@nenya.ms.mff.cuni.cz

školitel:
DOC. ING. FRANTIŠEK PLÁŠIL, CSC.
KSI MFF UK, Malostranské náměstí 25, 110 00 Prague 1
plasil@uivt.cas.cz,plasil@nenya.ms.mff.cuni.cz

obor studia:
Object Oriented Paradigm in Distributed Systems

Abstrakt

This paper presents todays industrial java component model - Enterprise JavaBeans (EJB). This component model is also getting more popular not only in industrial area, but education and research areas. Since the first version the EJB specification has been heavy evaluated, thus creators has started work on new version. The paper shows weak points of Enterprise JavaBeans component model with respect to generalization and enhancements. Further in the paper, there are proposed ways of solving these problems.

1. Introduction

Current software architectures has been moving towards larger software elements. These trends change software development from procedure based to component based design. One of merits of component design is creation of architecture description language (ADL) and component description languages (CDL) [2], which simplify description of software architecture and increase reusability of software elements (component). Such componet, together with its description, can be employed in different software project without changes in its source code. On the other side, there is a risk of insufficient description of a component, that can ends with unstable or non-reliable software system.

In next subsections, the brief overview of component models and Enterprise JavaBean (EJB) model are presented. Following section presents enhancements of the EJB architecture. The last section contains conclusion.

1.1. Component models

There is a number of academic component based software architectures. The well known are Writght [5], C2 [7], Rapide [8] and SOFA/DCUP [2]. All of them faces the same problem - how to describe component's behaviour, requirements on hosting environment and provided services. Component's functionality is provided by interfaces (set of methods or set of accepted and emitted events) together with some metainformation such protocol [3] specifing expected behavior. Industrial software world is also heading to components. The most widely used architectures are CORBA

[10], COM/DCOM [12], JavaBeans [13] and Enterprise JavaBeans [1]. Some of the have a one problem and one advantage compared to academic ones. The problem is, that they focused on specific part(s) of usage. The advantage is, that they have been fully implementated, not only as a "proof of the concept".

Unfortunatelly, these architectures are proprietary solution, thus there is no possibility to reuse components among them. Currently OMG [11] is trying to incorporate other component models (EJB) to its component model and create mapping to another component architecuters.

1.2. Enterprise JavaBean component model

The Enterprise Java Beans (EJB) [1] architecture is the one of the commercial component architectures. The EJB architecture is tightly bound to the Java language and environment. EJB is based on the concept of EJB container, which provides services (transactions, persistence, security) to inserted (deployed) components (beans) during runtime. Also, the EJB container manages the life of the deployed beans. The EJB specification introduces two phases of a bean s life: deployment and runtime. At the beginning of the deployment, the EJB container creates wrapping code to contacting the provided services (transaction, persistence, and security) by some of the business methods of the Remote interface; the wrapping is specified by the Bean s DeploymentDescriptor. Then, the bean is inserted (deployed) into the EJB container. At runtime, the wrapping code and the bean code are executed as a part of the EJB container. This way, the functionality of a bean (at its Remote Interface) can be parametrized by its DeploymentDescriptor, i.e., without the bean s object code involvement.

2. Enhancements of the EJB component model

Current EJB component model is focused on specific area of usage - business components, which requires transaction processing and persistent behavior. For simple and middle-complex applications, this seems to be sufficient, but how to provide more complex services to the components, how to ensure, that the component deployed in one place and rely on local services, will be connected to the services with the same behavior? Compared to the C2 [7] or SOFA/DCUP [2] the EJB's capabilities are limited. The goal of this paper is to propose extension compatible with [2] [3] [6] [9] to the EJB component model.

2.1. Multiple interfaces

The EJB component serves only to one purpose, that is given by its interface. Thus the component cannot provides different levels of service (for example defined by security restrictions). Common solution to that is providing of multiple interfaces - componen offers different "set of services" to the client and these "set of services" can be base on type of the client (e.g. untrusted client cannot access interface for managing accounts).

Technically, an EJB component's home interface would contain new method(s) to traverse available interfaces. To be compatible with current specification, one interface will be default.

2.2. Asynchronous invocation

Synchronous method invocation mechanism is not efficient in situations requiring late delivery of results (e.g. mobile computing or long time computing - client computer can be disconnected from the application and connected later to get the results). That can be achived by asynchronous invocation implemented by Java Messaging System. Another usage are event-based systems, where interfaces are not given as a set of methods, but as a set of events. Then some componets acts as a sources of events and provides "channels" and some components acts as a listeners who are connected to the channels. Thus asynchronous invocation can bind more then one listener to one channel.

To add asynchronous invocation pattern to the EJB component model, the multiple interfaces are required. The basic idea is to add new interface for each type of accepted and/or provided event. This approach does not affect old EJB client, because these interfaces will be hidden to him.

2.3. Provides and Requires

Application is composed of a number of components. The bindings between components is given by provides and requires interfaces. When a component requires an interface, the system (in the EJB model, this is the EJB container) has to find component or a service which provides requested interface. Then these two interfaces are connected. If interfaces consists of methods, the binding between provides and requires is 1 to 1. If interfaces consists of events, the binding between provides and requires is 1 to N.

To accomplish that, the EJB model would enhance its component's deployment descriptor of provides and requires property (list of provided interfaces and list of required interfaces). During the deployment, this information will be processed by the EJB container to resolve all dependecies between deployed components.

2.4. Versioning and protocols

The EJB components are black-boxes and an functionality is given by theirs interfaces and theirs implementation is hidden. The EJB model also does not deal with different versions of components and compatibility between them (e.g. an old component is replaced with new one; then because of the new one can be implemented using different techniques, the problem of incompatibility can arise). The solution to that problem is usage of behavior protocols [3]. Behavior protocols describe the sequences of invocation on requires and provides interfaces of components. In [3] compatibility between behavior protocols is defined, thus the compatibility between versions of a component can be specified. The protocol of component would be stored in deployment descriptor and check only during component's deployment. There is also another approaches to versioning based on a number systems (e.g. 1.2 or 45), but they have not strong describing capability. Another approach is to used lattices as propesed in [4].

2.5. Generic EJB container extension

The EJB container is the life-space for all EJB applications. It manages component's lifecycle and provides services for it. Service can change the behavior of component in two ways. The first one change the lifecycle management and/or invocation mechanism using aspect oriented programing or wrapper technique. In this case, components have limited access to service, but this service is transparent for component itsel and client too (e.g. security and persistent service is handled transparently to component). The second one is to provide interface to the services (e.g. transaction service is provided by interface to transaction object). The problem is, that only fixed set of services is given by the EJB specification - transactions, security and persistence. Behavior of these services is described in the EJB specification and should be the same for all containers.

But how can the new service should be added and should be specified? There could be specialized "service component" with specified priviledges to access internal structures of the container (necessary for persistence). Such component will have provides interfaces for components and requires interfaces, which could be binded to provided interfaces by another service component. Service will also be described by protocol. Weak point of this approach is access to internal structures of the container. Another way is to insert service's code into the container and provides interface and protocol to components.

3. Conclusion

The Sun Microsystem's Java Specification Request asks for specification proposal for new version of EJB model. This paper presents extensions to the EJB component model and problems with the theirs incorporation to the current EJB specification. These extensions will be futher evaluated.

References

[1] Sun Microsystems, "Enterprise JavaBean specification", http://java.sun.com/ejb/

[2] Plasil, F., Balek, D., Janecek, R., "SOFA/DCUP Architecture for Component Trading and DynamicUpdating.", In Proceedings of ICCDS '98, Annapolis, IEEE CS, 1998, pp. 43-52.

[3] Frantisek Plasil, Stanislav Visnovský, Miloslav Besta, "Behavioral Protocols and Components", TOOLS 1999

[4] Jaroslav Gergic, "Versioning in SOFA/DCUP", Charles university, master theses 1999

[5] Allen, R. J., "A Formal Approach to Software Architecture." Ph.D. Thesis, School of Computer Science,Carnegie Mellon University, Pittsburgh, 1997

[6] Ralph Melton and David Garlan, "Architectural Unification", School of Computer Science,Carnegie Mellon University, Pittsburgh, 1997

[7] Medvidovic Nenad, "The C2 style", http://www.ics.uci.edu/pub/arch/c2.html

[8] Luckham D., "Rapide project", http://pavg.stanford.edu/rapide/rapide.html

[9] David C. Luckhama, James Vera, Sigurd Meldal, "Three concepts of System Architecture", Stanford University, 1995

[10] Object Management Group, "CORBA Component model" http://www.omg.com

[11] "Object Management Group" http://www.omg.com

[12] Microsoft "COM/DCOM" http://www.microsoft.com/com

[13] Sun Microsystems "JavaBeans" http://java.sun.com/beans "Java Specification Request" http://java.sun.com/aboutJava/communityprocess/jsr.html

Decision support in medicine using electronic patient records

doktorand:
ING. PETR HANZLÍČEK
EuroMISE Centrum UK a AV ČR, Pod vodárenskou věží 2, 180 00, Praha 8
hanzlicek@euromise.cz

školitel:
PROF. RNDR. JANA ZVÁROVÁ, DRSC.
EuroMISE Centrum UK a AV ČR, Pod vodárenskou věží 2, 180 00, Praha 8
zvarova@euromise.cz

obor studia:
Teoretická informatika

1. Overview of HIS in the Czech Republic

Health Information System is a system which covers all areas of systematic entering, storing and processing of the data in a hospital. Creating of Health Information System demands challenging labor of specialized teams (physicians), analysts, programmers and other professions. Introduction and operation of such a complex system is expensive (hundreds of thousands - millions of USD) and its lifetime is 6-12 years. The environment of the system often rapidly changes (hospital infrastructure, legislation), increases volume and quality of processed data, new technologies appear. That's why the right selection of appropriate HIS is so important.

In the Czech Republic there are 207 hospitals (including 23 large regional or teaching hospitals with more than 1000 of beds) directly managed by the Ministry of Health. Most of the other ones are small hospitals which are part of governmental property. Only 12 hospitals are private ones belonging to church or companies. Nowadays there are about 40 hospitals with full Hospital Information System (HIS) consisting of following modules:

- patient management system
- patient care system
- hospital management system
- clinical information and reseach system.

Unfortunately in the rest of hospitals there are only parts of HIS, mainly concerned with hospital accounting system. Just several hospitals (about sixty) are running isolated modules of HIS without interaction. Here are some examples:

- finance and billing payment from insurance companies
- laboratory information system

- resource management and staff scheduling
- cathering
- transport and pharmacy.

In several hospitals there are also clinical modules for patient management systems but only for isolated units, e.g. surgery, hematology, stomatology, gynaecology.

The great problem to introduce HIS in the Czech Republic is a great deficit of financial resources for investment. For example to purchase HIS hardware and software for a large hospital with ten departments means the investment about 8 millions of USD. The investment is not possible to realize without a state financial support. Also the choice of HIS for the large hospitals is a difficult problem because of the small number of offers from companies. The main difficulties concern following differences: language, law, accounting system, interface system of insurance companies, interface system for statistics, national standards for health informatics and traditions of Czech physicians. The most of foreign companies seek for the first HIS contract with the aim to test their software in national conditions. By this way the large hospitals are in fact laboratories for software development with the consequence that the final solution and HIS implementation is postponed to infinity. Nowadays 8 large Czech hospitals are in such a situation and no hospital has quite completed HIS.

We can state that specification of HIS in the Czech Republic is following: Database environment: ORACLE (the most frequent), INFORMIX, PROGRESS (for small applications), MUMPS (rare). Trading names of HIS software in the Czech Republic are: SMS (USA), AMIS, SAS, LOGIS, STAPRO, SIEMENS NIXDORF (Germany), PCS, APP, SYSTEMA (Austria).

The first experience showed difficulties in training of health personnel, problems with national differencies lasting 3 years and finally the transfer to Czech laboratory information system from Ostrasoft company.

Smaller regional hospitals with hundreds of beds are in better situation. Here the investment for HIS is about 1 million of USD and there is large choice of software. The above stated number of local and foreign companies can be enlarged upon ten companies, HiComp Systems Brno, STEINER, DIALOG Brno etc. Removing of the national differencies of HIS is for a small hospital easier and more successful. There are 40 small hospitals with working HIS in the following cities, e.g. Breclav, Uherske Hradiste, Ceska Lipa.

The project of building HIS in the Czech Republic is very topical [1]. First of all, to finance HIS is a difficult task because of insufficient payments from insurance companies. On the other side, a working HIS is the main assumption for the high rentability of a hospital due to possible savings of one third of expenses. From the point of view of the Ministry of Health the main problem is a chaotic development of HIS without considering basic principles of HIS unification in the Czech Republic. All HIS should satisfy the following standards:

- methods and standards of the National Information System that is quaranted by the Institute of Health Information and Statistics of the Czech Republic (data for health yearbooks, OECD and WHO);
- data standards for patient data transfer among information systems of health care providers;
- national codes of laboratory items for sharing of the results of measurements between physicians and other users;
- obligatory interface for statistics;
- obligatory interface for health insurance;
- obligatory interface for Financial Offices;

- recommendation of the Ministry of Health on data protection;
- data standards for communication with governmental offices.

Another problem is connecting HIS into the national health data network, in fact to their virtual private nets for running health registers like waiting list for organs transplantation, register of cardiovascular interventions, national oncological register, register of unwanted blood donors. Further problem is monitoring of costing and budgeting of large hospitals directly managed by the Ministry of Health. There are being run first pilot projects implementing electronic (chip) health cards for citizens in a town of Litomerice [2]. These cards are also part of HIS in these projects. However, this is the nowadays concept of the Ministry of Health in the Czech Republic.

2. Multimedia electronic patient record ORCA

The project I4C of the 4th Framework Programme (1996 - 1998) was carried out for the further advancement of cardiac care. It was focused on clinical applications and its main goals were as follows.

- integrated access to data wherever stored;
- support of evidence-based care by remote electronic consultation and peer review;
- more comprehensive and more consistent recording of patient data, images, videos and biosignals, all combined in a multiple patient record.

With the support of the I4C project the new approach for multimedia electronic patient record has been developed.

Multimedia patient record ORCA (Open Record for CAre) is a system that integrates a possibility of structured patient data entry including history, medication, symptoms and more with multimedia objects as ECG, angiography or laboratory data [3]. Data can be entered either using prepared custom forms for special purposes or directly using a knowledge tree. Structured data facilitate statistical proccesing and translation of entered data into other languages. When it is impossible to enter determined facts using information in a knowledge tree, Orca provides a possibility to insert free text entry into the patient record. However, the prefered way is the structured data entry. Current version of Orca includes a knowledge tree and user interface translation into eight languages, including Czech, translated at the EuroMISE center (Orca's menus and user interface) and University hospital in Prague (Orca knowledge base). Translated versions were then returned to the project coordinator. The Orca system uses client-server model, which gives good performance, security and scaleability. For small installations like general practicioner's office, Orca can be run on a single computer with Windows95 using locally installed Interbase SQL server. The project I4C-TripleC intends to validate integrated workstations and ORCA system in three hospital in Central Europe (Prague, Bratislava and Caslav) to support the continuity of cardiac care.

The system will be used for cardiology patient data collecting with the aim to test functionality and usability of the system in specific conditions of our countries. Collected data then will be analyzed during research part of project using many tools and methods.

In our case the aim of the research is to search signs and symptoms relevant for decision making in the defined medical problem. The decision problem can be described as it follows. Our diagnostic hypothesis concerns a small disjunct set of decision variables Y, which we call dependent variables. Our task is to find a nonredundant set of symptom variables X called set of independent variables, such that knowledge of values for X makes it possible to estimate with high credibility values in Y. Each variable in X has its own weight describing cost of getting values of variable. We can figure this weight as a cost of investigation, e.g. painful investigation will have the height weight. In the process of searching of variables in X we should minimize the total weight of all variables obtained. The approach is based on methods of information theory, described in [4], [5], [6], [7], [8].

References

[1] Zámečník M: Overview of hospital information systems in the Czech Republic. Physician and Technology 1-2, 1997 (in Czech)

[2] Neuwirt K.: Project Macha - Study on health cards use in Litomerice. Physician and Technology 5, 1998 (in Czech)

[3] Pierik F.H., van Ginneken A.M., Timmers T., Stam H., Weber R.F.: Restructuring routinely collected patient data: ORCA applied to andrology. Yearbook of Medical Informatics 98, Schattauer, Stuttgart 1998

[4] Vajda I. : On the f-divergence and singularity of probability measures. Periodica Math Hung 2, 1972, 223-234

[5] Vajda I. : Theory of statistical inference and information, Kluwer, Dordrecht, 1989

[6] Zvárová J., Studený M., Preiss J. : On extracting relevant information from medical data. MEDINFO 96 proceedings, J. Brender et al., Amsterdam : IOS Press, 1996, 649-653

[7] Zvárová J., Studený M. : Information theoretical approach to constitution and reduction of medical data. Int J Med Inf 1997 Jun;45(1-2), 65-74

[8] Zvárová J., Tomečková M., Štefek M., Boudík F., Zára K. : Decision support and data analysis tools for risk assessment in primary preventive study of atherosclerosis. MEDINFO 97 proceedings, C. Pappas et al., Amsterdam : IOS Press, 1997, 625-628

Transformace a regrese

doktorand:
RNDr. Jan Klaschka

ÚI AV ČR, Pod vodárenskou věží 2, 182 07 Praha 8

klaschka@uivt.cas.cz

školitel:
Prof. RNDr. Radim Jiroušek, DrSc.

VŠE, LISP, nám. W. Churchilla 4, 130 67 Praha 3

radim@vse.cz

obor studia:
Operarační výzkum

Abstrakt

Příspěvek se nezabývá regresní analýzou transformovaných veličin, nýbrž regresí ve smyslu *návratu k dřívějšímu vývojovému stádiu*, s níž se autor setkal při studiu transformací zachovávajících tzv. kardinální regularitu struktur měření změny.

1. Úvod

Doktorand má být mladý. Ne každý však skutečně je. U doktoranda ve středním věku hrozí reálné nebezpečí, že bude obtěžovat vzpomínkami. Mladý doktorand nic takového dělat nemůže, protože má celý život před sebou, a tedy evidentně nemá nač vzpomínat. Pro úplnost dodejme, že není třeba příliš se obávat eventuálního starého doktoranda – může toho mít hodně za sebou, ale nejspíš mu nebude sloužit paměť.

Autor překročil letos na podzim práh středního věku. Čtenáři, těš se.

2. Výchozí předpoklady

V dětství jsem bydlel v budově internátu jisté školy, a k mým nejmilejším místnostem v domě patřila účtárna. Svátkem pro mě bylo, když jsem někdy směl místo paní účetní sčítat na stroji Nisa sloupce "Má dáti" a "Dal". Na rozdíl od většiny vrstevníků jsem také rád listoval v Korděmském [1] a Pionýrské olympiádě důvtipu [2].

Není divu, že si mě v mých patnácti či šestnácti vyhlédl maturant a pozdější student MFF UK Richard K., který k smrti rád vyprávěl o ordinálních číslech, nerozhodnutelných větách, bijekci množiny na vlastní podmnožinu, Russelově paradoxu, neměřitelných množinách a dalších divech. Mnoho jiných potenciálních obětí mu cca patnáctitisícové Mariánské Lázně nenabízely – RNDr. Milan Studený, CSc. (ÚTIA AV ČR a LISP VŠE) dle mého odhadu tehdy ještě chodil v námořnickém oblečku. Zvěstem, které Richard K. hlásal, jsem sice zpočátku naslouchal s jistou nedůvěrou, ale nakonec jsem se s nimi sžil natolik, že mi nezbylo, než také studovat na MFF.

Moje curriculum na fakultě pohříchu neobsahovalo žádný kurz teorie množin nebo logiky, ale i v analýze byla příležitost pokochat se třeba spojitým zobrazením úsečky na čtverec a podobnými bizarnostmi, které se ve světě normálního smrtelníka nevyskytují. Z "kouzel" tohoto druhu se mi snad nejvíce líbila konstrukce funkcí $f : R \to R$ (kde R je množina všech reálných čísel), které jsou aditivní, tj. splňují tzv. Cauchyovu funkcionální rovnici

$$f(x+y) = f(x) + f(y) \quad x, y \in R, \tag{1}$$

ale nejsou tvaru $f(x) = a\ x$, kde a je reálná konstanta. Pro toho, kdo to snad nezná, stručně rekapituluji: Reálná přímka R se vezme jako vektorový prostor *nad tělesem racionálních čísel.* Tento vektorový prostor má (nespočetnou) bázi B. Funkce f se na B definuje libovolně a na R se rozšíří jako lineární zobrazení uvedeného vektorového prostoru do sebe. Tedy, konstrukce to je a není – báze B se nedá (v nějakém rozumném smyslu) sestrojit, ale její existence je důsledkem axiomu výběru. Je známo (viz např. skripta MFF UK [3] nebo článek [4]), že řešení rovnice (1) s výjimkou funkcí $f(x) = a\ x$ jsou "divoké" funkce: jsou vesměs lebesgueovsky neměřitelné, nejsou nikde spojité, na žádném otevřeném intervalu nejsou zdola ani shora omezené.

Z pozorování podobných úkazů jsem se však mohl těšit jen v první polovině studia – později na specializaci matematická statistika byla matematika daleko pragmatičtější. A v praxi medicínského statistika jsem pak už měl úplně jiné starosti – třeba jak se před lékařem neuřeknout, že aritmetický průměr je náhodná veličina, a neztratit jeho důvěru. Bylo to jasné: S mládím odcházejí do nenávratna i "divoké" funkce a další rekvizity. Už nikdy je neuvidím.

3. Transformace zachovávající regularitu

Koncem osmdesátých let jsem se při práci ve Výzkumném ústavu psychiatrickém setkal s velmi podezřelou transformací dat, tzv. indexem terapeutické účinnosti, vulgo Rakúsovým indexem. Rakúsův index má vyjadřovat velikost zlepšení, ke kterému u pacienta trpícího schizofrenií došlo mezi dvěma vyšetřeními. Vypočte se z dílčích skórů udávajících tíži jednotlivých příznaků při prvém a druhém vyšetření. Konkrétní podobu vzorce (viz [5]) ponechme stranou – podstatné je, že se Rakúsův index chová nelogicky, např. umožňuje (alespoň teoreticky), aby postupné změny hodnocené vesměs jako zlepšení dávaly dohromady změnu hodnocenou jako zhoršení.

Jako určitá odpověď na otázku, co psychiatrickým výzkumníkům při návrhu indexů určených k hodnocení zlepšení (nebo zhoršení) odůvodněně přikázat či zakázat, vzniklo obecné kritérium tzv. *regularity struktur měření změny* a několik jeho konkrétních instancí, konkrétně *kardinální, ordinální* a *slabá ordinální regularita.* Problematice regularity jsem se pak věnoval ne sice souvisle, ale dlouho, a učinil jsem z ní také téma své disertace.

Strukturou měření změny rozumím trojici $\langle S, G, H\rangle$, kde $S \neq \emptyset$ je množina *stavů*, $G \subseteq S \times S$ ($G \neq \emptyset$) je množina možných *přechodů* mezi stavy (první souřadnice reprezentuje *počáteční stav*, druhá *konečný stav*) a $H : G \to R$ je tzv. *index změny.* Jedna z možných definic kardinální regularity (ostatním typům regularity se věnovat nebudeme) vypadá následovně. Řekneme, že $\langle S, G, H\rangle$ je *kardinálně regulární*, pokud pro každé dvě n-tice ($n \geq 1$) přechodů $(s_1, t_1), \ldots, (s_n, t_n) \in G$ a $(s'_1, t'_1), \ldots, (s'_n, t'_n) \in G$ takové, že $s_1, \ldots, s_n$ je permutace $s'_1, \ldots, s'_n$ a $t_1, \ldots, t_n$ je permutace $t'_1, \ldots, t'_n$, platí

$$\sum_{i=1}^{n} H(s_i, t_i) = \sum_{i=1}^{n} H(s'_i, t'_i). \tag{2}$$

Kardinální regularita je, jak snadno nahlédneme, vlastnost invariantní vůči lineárním transformacím: Nechť $T : R \to R$ je tvaru $T(x) = a\ x + b$, kde a a b jsou reálné konstanty. Je-li struktura měření změny $\langle S, G, H\rangle$ kardinálně regulární, platí, kdykoli je definicí požadována rovnost (2), také

$$\sum_{i=1}^{n} T\big(H(s_i, t_i)\big) = \sum_{i=1}^{n} T\big(H(s'_i, t'_i)\big), \tag{3}$$

takže $\langle S, G, T(H)\rangle$ je kardinálně regulární.

Jsou však lineární transformace *jediné* transformace zachovávající kardinální regularitu? (Rozumějme: Je transformace T lineární, pokud pro *každou* kardinálně regulární strukturu měření změny $\langle S, G, H\rangle$ platí, že také $\langle S, G, T(H)\rangle$ je kardinálně regulární? Pro *konkrétní* kardinálně regulární $\langle S, G, H\rangle$ nezávisí kardinální regularita $\langle S, G, T(H)\rangle$ vůbec na hodnotách $T(x)$ v takových bodech $x \in R$, že H hodnoty x nikde nenabývá.)

Moje odpověď v "hrubé" verzi disertace byla kladná. Nikoli však správná. Když jsem se dal do psaní důkazu toho, co jsem měl dlouho za naprosto zřejmé (příklad kardinálně regulární struktury měření $\langle S, G, H\rangle$ sestrojené "na míru" pro nelineární transformaci T tak, aby struktura $\langle S, G, T(H)\rangle$ kardinálně regulární nebyla, jsem *živě viděl*), čekalo mě nemalé překvapení.

Fakt, že lineární transformace T zachovává kardinální regularitu, vyplývá z toho, že kdykoli platí (2), platí také (3). Jinými slovy, transformace T zachovává kardinální regularitu, pokud platí implikace

$$\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i \Rightarrow \sum_{i=1}^{n} T(x_i) = \sum_{i=1}^{n} T(y_i) \quad x_i, y_i \in R, i = 1, \ldots, n (n \geq 1). \tag{4}$$

Pokud ještě není jasné, kam jsme se to dostali, pak jistě bude, zjistíme-li, že (4) je ekvivalentní funkcionální rovnici

$$T(x) + T(y) = T(z) + T(x + y - z) \quad x, y, z \in R. \tag{5}$$

Nebo po lopatě: Funkce T je řešením (4), resp. (5), právě když pro nějaké $c \in R$ platí,

$$T(x) = c + f(x) \quad x \in R,$$

kde $f : R \to R$ je řešení (1).

Funkce zachovávající kardinální regularitu jsou tedy o konstantu posunuté aditivní funkce. Všechny, včetně "divokých" – neměřitelných, atd. Dobrý den, to jsou k nám hosti!

4. Jiné transformace

Lhal bych, kdybych tvrdil, že mi nečekané setkání s aditivními funkcemi způsobilo jen samou radost. Zde jsou některé problémy, které s sebou přináší.

- Kritérium kardinální regularity je určeno pro případ, kdy index změny má být veličinou intervalového typu. (O typech dat viz např. [6] z pohledu statistiky, popř. [7] v kontextu teorie měření.) Kdyby kardinální regularitu zachovávaly *právě jen* lineární transformace, dalo by se to chápat jako dílčí stvrzení toho, že kritérium je pro daný typ dat dobře navrženo. A naopak, když ne.

- Neměřitelná řešení rovnice (5), jakkoli zachovávají kardinální regularitu, naprosto mění (alespoň v některých případech) obsah "zpráv", které index změny podává o tom, který přechod mezi stavy je více, a který méně přijatelný.

- Ordinální regularitu, která je slabší vlastností struktur měření změny než kardinální regularita, zachovávají konstantní a ryze monotónní transformace. Slabou ordinální regularitu, která je ještě slabší vlastností, zachovávají neryze monotónní transformace. Kdyby kardinální regularitu zachovávaly právě jen lineární transformace, platilo by alespoň pro dané tři typy regularity, že čím silnější vlastnost, tím menší množina transformací, které ji zachovávají. S nemonotónními transformacemi zachovávajícími kardinální regularitu se tato hierarchie hroutí.

- Komplikuje se teorie měření neregularity: Míra neregularity určitého typu je zobrazení Λ přiřazující každé struktuře měření změny $\langle S, G, H\rangle$ nezáporné číslo $\Lambda(S, G, H)$, přičemž $\Lambda(S, G, H) = 0$, právě když $\langle S, G, H\rangle$ je regulární. Pro každý typ regularity potřebujeme mít definovány třídy transformací, vůči kterým by se míry neregularity měly "chovat slušně" (být invariantní či nerůst). Jakkoli se zdá přirozené založit definici těchto tříd na zachování regularity, bude ve skutečnosti třeba postupovat složitěji, protože není dost dobře možné zavést takové rozumné míry neregularity, které by se "slušně chovaly" i vzhledem k nelineárním aditivním transformacím.

K dalším úvahám potřebujeme určitý aparát. O pravděpodobnostní míře P budeme mluvit jako o *jednoduchém (pravděpodobnostním) rozdělení* na množině $A \neq \emptyset$, pokud je definována pravděpodobnost $P(\{a\})$ pro každou jednoprvkovou podmnožinu $\{a\} \subseteq A$ a pokud existuje taková *konečná* podmnožina $A_0 \subseteq A$, že $P(A_0) = 1$. Jednoduché rozdělení na množině A je *empirické*, pokud pro každou množinu $B \subseteq A$ je pravděpodobnost $P(B)$ racionální číslo. Množinu všech jednoduchých, resp. jednoduchých empirických rozdělení na množině A označíme $\mathcal{P}_A$, resp. $\mathcal{P}^*_A$.

Je-li $\langle S, G, H\rangle$ struktura měření změny, budeme o rozdělení $P \in \mathcal{P}_G$, resp. $P \in \mathcal{P}^*_G$ mluvit jako o *rozdělení přechodů*, resp. *empirickém rozdělení přechodů*. Rozdělení přechodů indukuje marginální rozdělení P_1 a P_2 (tzv. *rozdělení počátečních* a *konečných stavů*) definovaná vztahy

$$P_1(A) = P\big((A \times S) \cap G\big), \quad P_2(A) = P\big((S \times A) \cap G\big) \quad A \subseteq S.$$

Pokud pro $P, Q \in \mathcal{P}_G$ je $P_1 = Q_1$ a $P_2 = Q_2$, řekneme, že *P a Q jsou ekvimarginální.*

Na index změny lze při daném rozdělení přechodů pohlížet jako na náhodnou veličinu. *Rozdělení funkce H při rozdělení přechodů* $P \in \mathcal{P}_G$ je rozdělení $PH^{-1} \in \mathcal{P}_R$ definované vztahem

$$PH^{-1}(A) = P\big(\{g \in G; H(g) \in A\}\big) \quad A \subseteq R.$$

Střední hodnota H při rozdělení $P \in \mathcal{P}_G$ je dána jako

$$E_P H = \sum_{g \in G} P(g)H(g).$$

Při daném rozdělení $P \in \mathcal{P}_R$ lze také transformaci $T : R \to R$ chápat jako náhodnou veličinu s rozdělením $PT^{-1} \in \mathcal{P}_R$. Pro střední hodnotu $E_P T(H)$ transformovaného indexu změny $T(H)$ při rozdělení přechodů $P \in \mathcal{P}_G$ platí

$$E_P T(H) = E_{PH^{-1}} T,$$

a speciálně střední hodnotu $E_P H$ lze vyjádřit jako

$$E_P H = E_{PH^{-1}} id,$$

kde id je identická transformace (tj. $id(x) = x$ pro $x \in R$).

Kardinální regularitu můžeme (vedle způsobu již uvedeného) ekvivalentně definovat také takto: Řekneme, že struktura měření změny $\langle S, G, H\rangle$ je kardinálně regulární, pokud pro každá dvě rozdělení $P, Q \in \mathcal{P}^*_G$, která jsou ekvimarginální, platí $E_P H = E_Q H$.

V následujících odstavcích budou uvedeny některé nápady, jak buďto pozměnit definici kardinální regularity (či regularity obecně) tak, aby neměřitelné aditivní funkce nadále kardinální regularitu nezachovávaly, nebo jak definovat transformace "použitelné" při měření neregularity, aby uvedené "divoké" funkce "vypadly ze hry". Nápady jsou doprovázeny námitkami, které je částečně nebo úplně diskvalifikují. Většina těchto nápadů a námitek padla letos na přelomu srpna a září v několikadenní e-mailové diskusi s kolegou RNDr. Petrem Savickým, CSc. během jeho pobytu v Dortmundu. Srdečně mu děkuji.

Nápad: Skutečnost, že nelineární aditivní funkce zachovávají kardinální regularitu, bude souviset s tím, že v definici kardinální regularity se mluví jen o *empirických* ekvimarginálních rozděleních. Kdybychom místo $\mathcal{P}^*_G$ měli v definici $\mathcal{P}_G$, zachovávaly by kardinální regularitu nejspíše jen lineární transformace.

Námitka: I když to vypadá na první pohled pravděpodobně, není tomu tak. Podle obou definic, stávající i upravené, by byly kardinálně regulární tytéž struktury měření změny – úpravou definice by se tudíž nezměnila ani množina transformací, které kardinální regularitu zachovávají. "Může za to" fakt, že pokud $P, Q \in \mathcal{P}_G$ jsou ekvimarginální, lze dvojici (P, Q) vyjádřit jako směs ekvimarginálních dvojic *empirických* rozdělení přechodů (P^i, Q^i), $i = 1, \ldots, n$ $(n \geq 1)$, tj. pro nějaké kladné reálné koeficienty $\alpha_1, \ldots, \alpha_n$ takové, že $\sum_{i=1}^n \alpha_i = 1$, platí (přičemž smysl operací je zřejmý)

$$(P, Q) = \sum_{i=1}^{n} \alpha_i (P^i, Q^i).$$

Nápad: Nezachovávaly by kardinální regularitu pouze lineární transformace, kdybychom v definici regularity nahradili *jednoduchá* rozdělení širším systémem pravděpodobnostních měr?

Námitka: Možná, ale kdybychom zřejmým způsobem definovali transformace zachovávající kardinální regularitu *konečných* struktur měření změny, tj. takových struktur měření změny $\langle S, G, H \rangle$, že množina G je konečná, nelineární aditivní transformace by mezi nimi opět byly.

Nápad: V teorii kolem měření neregularity bychom se jednoduše zbavili nelineárních řešení rovnice (5), kdybychom v definici tříd transformací, vůči kterým se mají míry neregularity "dobře chovat", mluvili o *měřitelných* transformacích splňujících to a to. Měřitelnost přece představuje velmi mírné omezení a k aplikaci žádné neměřitelné transformace v praxi stejně nikdy nedojde.

Námitka: Požadavek lebesgueovské či borelovské měřitelnosti by nezapadal organicky do celé práce o regularitě a mírách jejího porušení. V souvislosti s pravděpodobností nepadne o měřitelnosti slovo, protože pracujeme s jednoduchými rozděleními, a tedy máme *všechny* množiny měřitelné.

Nápad: Všechna řešení rovnice (5) se chovají na množině všech racionálních čísel jako lineární funkce. Kdybychom do definice struktury měření změny přidali podmínku, že index změny musí nabývat jen racionálních hodnot (a trochu "upravili okolí"), bylo by po problémech. Indexy změny si vymýšlejí lékaři, a jiná než racionální čísla k tomu stejně nepoužívají.

Námitka: Kdyby někdo chtěl skórovat stavy pacientů $1, 2, \pi, 4, \ldots$, asi by byl důvod rozmlouvat mu to jako pošetilost. Ale zakazovat třeba definici indexu změny pomocí vzorců s odmocninami?

Nápad: Transformace $T : R \to R$ zachovává kardinální regularitu právě tehdy, když pro všechny dvojice měr $P, Q \in \mathcal{P}_R^*$ platí implikace

$$E_P id = E_Q id \Rightarrow E_P T = E_Q T. \tag{6}$$

Nezbavili bychom se nelineárních řešení rovnice (5), kdybychom žádali "slušné chování" měr neregularity vůči takovým transformacím, že implikace (6) platí pro všechny dvojice měr $P, Q \in \mathcal{P}_R$?

Námitka: Ano takovou podmínku by splňovaly pouze lineární transformace. Všimněme si ale, že zesílená podmínka není v těsné souvislosti se zachováním kardinální regularity. Jde o to, že každou dvojici měr $P, Q \in \mathcal{P}_R^*$ takovou, že $E_P id = E_Q id$, dostaneme jako $P = P'H^{-1}$ a $Q = Q'H^{-1}$ pro nějakou kardinálně regulární strukturu měření změny $\langle S, G, H \rangle$ a nějaká ekvimarginální rozdělení přechodů $P', Q' \in \mathcal{P}_G^*$, ale některé dvojice $P, Q \in \mathcal{P}_R$ takové, že $E_P id = E_Q id$, v obdobném vztahu s žádnou kardinálně regulární strukturou měření změny $\langle S, G, H \rangle$ a žádnou dvojicí $P', Q' \in \mathcal{P}_G$ nejsou. Navíc by asi bylo lepší, aby definice byly "robustní" v tom smyslu, že povedou k témuž, ať v nich hovoříme jen o empirických rozděleních, nebo ne.

Nápad: Obecná definice regularity (kterou si dovolím jako takovou neuvést) pracuje s abstraktní binární relací $\prec$ na $\mathcal{P}_R$ a s relací $\sim$ odvozenou od $\prec$ tak, že $P \sim Q$, pokud není ani $P \prec Q$, ani $Q \prec P$. Konkrétní typy regularity dostaneme specifikací relace $\prec$. Pro kardinální regularitu má $P \prec Q$ význam $E_P id < E_Q id$ a $P \sim Q$ znamená $E_P id = E_Q id$. Relace $\prec$ se vlastně v definici regularity vyskytuje jen nepřímo – prostřednictvím relace $\sim$. Zatímco u ordinální a slabé ordinální regularity tato okolnost nemá žádné zvláštní důsledky, v případě kardinální regularity ano. Kardinální regularita se dá definovat pomocí rovnosti součtů jistých hodnot indexu změny, bez jakékoli zmínky o *nerovnostech* mezi součty. Pak se nelze divit, že transformace zachovávající kardinální regularitu mají blízko k homomorfismům aditivní grupy $\langle R, +\rangle$ do sebe, tedy řešením Cauchyovy funkcionální rovnice – včetně takových řešení, která nejsou homomorfismy *uspořádané* aditivní grupy $\langle R, +, <\rangle$ do sebe nebo do $\langle R, +, >\rangle$.

Z této úvahy vychází definice *přípustné transformace* a *slabě přípustné transformace*. Přípustné transformace mají míry neregularity zachovávat, slabě přípustné je nemají zvětšovat. Uveďme definice přípustné a slabě přípustné transformace pro případ kardinální regularity (resp. měr kardinální neregularity).

Transformace $T : R \to R$ je *přímo*, resp. nepřímo slabě přípustná, pokud pro každou dvojici $P, Q \in \mathcal{P}_R^*$ platí implikace

$$E_P id = E_Q id \Rightarrow E_P T = E_Q T,$$

a zároveň implikace

$$E_P id < E_Q id \Rightarrow E_P T \leq E_Q T,$$

resp.

$$E_P id < E_Q id \Rightarrow E_P T \geq E_Q T.$$

Transformace je *slabě přípustná*, pokud je přímo nebo nepřímo slabě přípustná. Transformace T je *přípustná*, pokud je slabě přípustná, a navíc platí pro $P, Q \in \mathcal{P}_R^*$ implikace

$$E_P id \neq E_Q id \Rightarrow E_P T \neq E_Q T.$$

Tentokrát dostáváme (podle obecné definice) pro všechny tři konkrétní typy regularity přípustné, resp. slabě přípustné transformace, "jaké potřebujeme": lineární kromě konstant, resp. všechny lineární pro kardinální typ, ryze monotónní, resp. ryze monotónní a konstantní pro ordinální typ, ryze monotónní, resp. neryze monotónní pro slabý ordinální typ. Mj. pro tyto typy regularity platí, že čím je regularita silnější vlastnost, tím jsou menší odpovídající množiny přípustných a slabě přípustných transformací. A konečně, množiny přípustných a slabě přípustných transformací pro kardinální, ordinální a slabý ordinální typ regularity se nezmění, nahradíme-li v definici množinu $\mathcal{P}_R^*$ množinou $\mathcal{P}_R$.

Námitka: Bylo by ještě lepší, kdybychom dovedli zaručit, že množiny přípustných a slabě přípustných transformací budou nezávislé na volbě systému měr ($\mathcal{P}_R^*$ nebo $\mathcal{P}_R$) v definici a budou v náležitých hierarchických vztazích nejen pro tři konkrétní typy regularity, ale obecně – pro všechny typy regularity (tj. specifikace relace $\prec$), které splňují nějaké rozumné postačující podmínky. Žádný takový výsledek ale v současnosti nemám.

A není nakonec celá ta snaha odvodit přípustné transformace z definice toho kterého typu regularity nesmyslná? Neměli bychom radši začít z jiného konce? Struktury měření změny $\langle S, G, H\rangle$ a $\langle S, G, T(H)\rangle$ by měly mít stejnou míru neregularity zejména tehdy, když (resp. proto, že) indexy změny H a $T(H)$ poskytují v podstatě tytéž informace. Přípustné transformace (otázku slabě přípustných transformací ponechme pro jednoduchost stranou) bychom pak mohli volit *a priori* – podle toho, jaký typ dat mají hodnoty indexu změny představovat (a jaká data lze tudíž považovat za v podstatě stejná). Tím bychom se také přiblížili pojetí typů dat a přípustných transformací běžnému v teorii měření (viz [7]). Co se týče vztahu přípustných transformací k zachování regularity, mohli bychom se pak spokojit s tím, že regularita struktury měření změny je invariantní vůči přípustným transformacím, a tedy jde – v terminologii teorie měření – o *smysluplnou vlastnost*.

To je provizorní konec přemítání o transformacích. Pokud čtenáři schází happy-end, omlouvám se a ujišťuji, že mně dvojnásob.

5. Perspektiva regrese

Vraťme se ale k překvapivému setkání po letech s Cauchyovou funkcionální rovnicí, kterou jsem naposledy viděl ve druhém ročníku na MFF UK. Čekal bych ho snad při teoretickém bádání v "čisté" matematice. Ale při práci na jakýchsi směrnicích pro zacházení s psychiatrickými škálami?

Instinkty statistika velí nevěřit, že to, co je příliš nepravděpodobné, přichází náhodou. Napadá mě, že třeba jsem svědkem vlastní *regrese* v některém z významů uvedených v encyklopedii [8]: *zpětný pochod, postup, návrat, ústup, úpadek,* popř. *návrat do některých z předchozích etap psychického vývoje.*

Pokud by tomu tak bylo, stáří by nemuselo být tak hrozné. K Nise bych se, pravda, vracel nerad, ale Korděmskij a Pionýrská olympiáda důvtipu na mě z knihovny pomrkávají vcelku přívětivě.

References

[1] B.A. Korděmskij, *Matematické prostocviky*, Mladá fronta, Praha, 1957.

[2] J. Karel, *Pionská olympiáda důvtipu*, Mladá fronta, Praha, 1963.

[3] J. Lukeš a kol., *Problémy z matematické analýzy*, SPN, Praha, 1972.

[4] A. Wilansky, "Additive functions", K. May (ed.), *Lectures on Calculus*, pp. 97–124, Holden-Day, San Francisco, 1967.

[5] A. Rakús et al., "Matematičeskije aspekty klinikofarmakokinetičeskoj predikcii effektivnosti terapii", *Žurnal nevropatologii i psichiatrii*, vol. 84, pp. 406–410, 1984.

[6] T. Havránek, *Statistika pro biologické a lékařské vědy*, Academia, Praha, 1993.

[7] F.S. Roberts, *Measurement Theory with Applications to Decissionmaking, Utility, and the Social Sciences*, Addison-Wesley, Reading, Massachusetts, 1977.

[8] *Malá československá encyklopedie*, sv. 5, ACADEMIA, Praha, 1986.

The joy of Joyce

doktorand:
ARNOŠT ŠTĚDRÝ
ÚI AV ČR, Pod vodárenskou věží 2, 182 07 Praha 8
arnost@uivt.cas.cz

školitel:
RNDR. JIŘÍ WIEDERMANN DRSC
ÚI AV ČR, Pod vodárenskou věží 2, 182 07 Praha 8
wieder@uivt.cas.cz

obor studia:

Abstrakt

V článku se popisuje vznik, vývoj, vybavení a současný stav ústavního výpočetního klusteru.

A lost long candle wandered up the sky from Mirus bazaar in search of funds for Mercer's hospital and broke, drooping, and sheda cluster *of violet but one white stars..*
James Joyce: Ulysses, Chapter Nausicca

1. Úvod

Žijeme v časech, kdy ceny počítačů klesají, stejně jako ceny síťového hardware. Doba, kdy si výrobu a prodej superpočítačů mohly dovolit pouze velké společnosti je nenávratně pryč. Milénium se zjevně odehraje ve znamení laciných superpočítačů s největší pravděpodobností založených na Beowulfu.

Tento dokument má být jakousi zprávou o zrodu ústavního superpočítače založeného na Beowulfu a zároveň i stručným průvodcem po jeho výrobě, správě a vlastním fungování. Autoři si nečiní nijakého nároku na definitivnost této zprávy, neboť svět Beowulfu prožívá prudký vzestup.

Na počátku je nutno zdůraznit, že Beowulf není žádná speciální topologie, nebo softwarový balík, ale pouze ideový princip jak stavět laciné superpočítače. Kolem Beowulfu existuje pochopitelně mnoho balíků, pomocných programů, či záplat systémového jádra, ale v konečném důsledku záleží vždy na tvůrci, po které z mnoha cest se vydá. V tomto smyslu je každý Beowulf unikátní stavba ve světě superpočítačů.

Jaké jsou základní charakteristiky superpočítačů Beowulfového typu?

- Jsou vyrobeny z běžně dostupných komponent. Neobsahují žádný zákaznický obvod ani jinou hardwarovou výstřednost.
- Nejčastěji jsou organizovány jako sestava jednoho či více serveru a mnoha bezdiskových uzlů navzájem propojených prostřednictvím ethernetových adaptérů, případně switchů.

- Softwarové vybavení je postaveno na volně dostupných produktech jako Linux, PVM, Mosix, MPICH, ...
- Díky stavebnicovému principu jsou snadno škálovatelné.

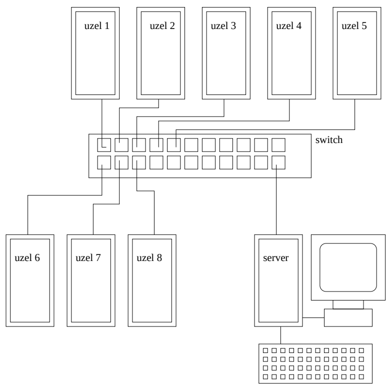
Famed was this Beowulf: far flew the boast of him, son of Scyld, in the Scandian lands. So becomes it a youth to quit him well with his father's friends, by fee and gift, that to aid him, aged, in after days, come warriors willing, should war draw nigh, liegemen loyal: by lauded deeds shall an earl have honor in every clan.

2. Technická realizace klusteru

Ústavní Beowulf Joyce je sestaven z 16 bezdiskových stanic a jednoho serveru. Konfigurace stanic je Celeron 433Mhz, 256MB paměti, server má paměti 390Mb. Celá sestava je propojena 100Mbitovým ethernetem prostřednictvím switche o celkové propustnosti 3.5Gbitů. Síťové karty uzlů jsou vybaveny bootovacím firmware. Disková kapacita celého klusteru je 10GB. Pořizovací cena celé sestavy se pohybovala kolem 500 000kč.

Bootování operačního systému popíšeme pouze v bodech, neboť se jedná o rutinní záležitost.

1. DHCP dotaz, kterým je nodu zděleno, kde se nachází jádro operačního systému.
2. Stažení jádra pomocí TFTP protokolu. Jádro musí být zkompilováno s podporou NFS, a kořenového zařízení na NFS. Obraz jádra musí být navíc označkován tak, aby jej bootovací firmware dokázal zavést do paměti.
3. Zavedení jádra, namountování kořenového zařízení via NFS, běžný start systému.



Obrázek 2: Schema klusteru Joyce

3. Mosad, Mosix, queue

Na počátku této sekce jsme v situaci, kdy již jednotlivé uzly nabootovaly svůj operační systém a my bychom rádi vytvořili transparentní prostředí pro spouštění a běh procesů. Transparentností v tomto případě míníme fakt, že běžný uživatel nevidí jednotlivé počítače, ale systém se mu jeví jako stoj jediný.

Toto prostředí by mělo sledovat zatížení jednotlivých uzlů a spouštět nové procesy na uzlech nevyužívaných. Rádi bychom, aby též disponovalo možnostmi zařazovat procesy do různých front dle jejich priority či systémových nároků, případně uvědomění uživatele prostřednictvím elektronické pošty o dokončení jeho úlohy. (Výhodu takového upozornění ocení provozovatelé obzvláště rozsáhlých výpočtů.)

Takovým správcem úloh je například GNU queue. Queue sestává z démonů queued běžících na každém stroji a sledujících zátěž procesoru, množství volné pamětí, počet právě běžících úloh, atd. Jednotlivé procesy jsou spouštěny pomocí klienta queue, který se nejprve informuje u příslušných démonu a na základě výsledků zvolí nejlepší procesor. Standardní vstup a výstup je přesměrován na hosta, ze kterého je klient queue spouštěn. Klient queue typicky nahrazuje shell Beowulfu, tedy uživatel je od celého plánování odstíněn.

Naprosto jiný přístup zvolili autoři projektu Mosix, kteří veškerou správu umístění a případné migrace procesu na jiný stroj vtělili do jádra systému. Mosix jde v transparentnosti tak daleko, že i pro většinu systémových nástroje je skryt hostitel daného procesu. Mosix navíc sleduje mnoho atributů jednotlivých procesorů a procesů a provádí migraci procesů mezi procesory k dosažení optimálního běhu.

4. MPICH, OpenMP, PVM

Máme nyní vytvořený systém, který nám umožňuje transparentní spouštění procesů po celém klusteru. Triviálně paraelizovatelné úlohy, tj. ty, které nevyžadují přílišnou komunikaci mezi procesy můžeme již v tomto okamžiku řešit. Rádi bychom ovšem naučili naše procesy komunikovat, tj, vytvořit pokud možno transparentní prostředí pro *message passing*.

Pro komunikaci mezi procesy se nejčastěji využívají dva balíky — PVM a MPI.

PVM je jednoduché prostředí sloužící k předávání zpráv mezi procesy a jejich vzájemné synchronizaci. tento systém byl původně vyvíjen pro až patologicky heterogenní sítě, takže většinu jeho vlastností zůstane u Beowulfu nevyužita. Přesto se jedná o nástroj nadmíru užitečný.

Poněkud novějšího data je MPI. Jeho nesporná výhoda je, že se jedná o systém podporovaný výrobci hardware, takže programy využívající MPI lze teoreticky spustit i na komerčních superpočítačích. V současné době existují dvě volně dostupné implementace MPI – MPICH a LAM.

Plánování rozvržení procesů na jednotlivé hosty není ani u jednoho z popsaných balíků příliš sofistikované. Proto jsme jej v našem případě nahradili klientským programem queue.

5. Budoucnost

V současné době chybí ústavnímu Beowolfu dobrý překladač paralelních verzí jazyků Fortran, C a C++. Testování a výběr vhodného nástroje bude probíhat v následujících týdnech. Zatím nejžhavějším kandidátem je produkt firmy Portland Group, Inc.

References

[1] Beowulf Underground http://beowulf-underground.org/

[2] Mosix http://www.mosix.cs.huji.ac.il/

[3] MPICH http://www-unix.mcs.anl.gov/mpi/mpich/

[4] PVM http://www.epm.ornl.gov/pvm

[5] Queue http://www.gnu.org/software/queue/queue.html

```c
#include <stdlib.h>
#include <stdio.h>
#include <pvm3.h>

#define NPROC   4

main(int argc, char **argv)
{
  register double lsum, width;
  double sum;
  register int intervals, i;
  int mytid, iproc, msgtag = 4;
  int tids[NPROC];  /* array of task ids */

  /* enroll in pvm */
  mytid = pvm_mytid();

  /* Join a group and, if I am the first instance,
     iproc=0, spawn more copies of myself
  */
  iproc = pvm_joingroup("pi");

  if (iproc == 0) {
    tids[0] = pvm_mytid();
    pvm_spawn("pvm_pi", &argv[1], 0, NULL, NPROC-1, &tids[1]);
  }
  /* make sure all processes are here */
  pvm_barrier("pi", NPROC);

  /* get the number of intervals */
  intervals = atoi(argv[1]);
  width = 1.0 / intervals;

  lsum = 0.0;
  for (i = iproc; i<intervals; i+=NPROC) {
    register double x = (i + 0.5) * width;
    lsum += 4.0 / (1.0 + x * x);
  }

  /* sum across the local results & scale by width */
  sum = lsum * width;
  pvm_reduce(PvmSum, &sum, 1, PVM_DOUBLE, msgtag, "pi", 0);

  /* have only the console PE print the result */
  if (iproc == 0) {
    printf("Estimation of pi is %f\n", sum);
  }

  /* Check program finished, leave group, exit pvm */
  pvm_barrier("pi", NPROC);
  pvm_lvgroup("pi");
  pvm_exit();
  return(0);
}
```

Obrázek 3: Příklad programu využívající PVM

```c
#include <stdlib.h>
#include <stdio.h>
#include <mpi.h>

main(int argc, char **argv)
{
  register double width;
  double sum, lsum;
  register int intervals, i;
  int nproc, iproc;
  MPI_Status status;

  if (MPI_Init(&argc, &argv) != MPI_SUCCESS) exit(1);
  MPI_Comm_size(MPI_COMM_WORLD, &nproc);
  MPI_Comm_rank(MPI_COMM_WORLD, &iproc);
  intervals = atoi(argv[1]);
  width = 1.0 / intervals;
  lsum = 0;
  for (i=iproc; i<intervals; i+=nproc) {
    register double x = (i + 0.5) * width;
    lsum += 4.0 / (1.0 + x * x);
  }
  lsum *= width;
  if (iproc != 0) {
    MPI_Send(&lbuf, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD);
  } else {
    sum = lsum;
    for (i=1; i<nproc; ++i) {
      MPI_Recv(&lbuf, 1, MPI_DOUBLE, MPI_ANY_SOURCE,
               MPI_ANY_TAG, MPI_COMM_WORLD, &status);
      sum += lsum;
    }
    printf("Estimation of pi is %f\n", sum);
  }
  MPI_Finalize();
  return(0);
}
```

Obrázek 4: Příklad programu požívající MPI

Trust region methods for large-scale optimization problems

doktorand:
Mgr. Ctirad Matonoha
Department of Numerical Mathematics, Faculty of Mathematics and Physics, Charles University of Prague, Malostranské náměstí 25, 118 00 Prague 1, Czech Republic.
matonoha@menza.mff.cuni.cz

školitel:
Doc. RNDr. Jan Zítko, CSc.
Department of Numerical Mathematics, Faculty of Mathematics and Physics, Charles University of Prague, Malostranské náměstí 25, 118 00 Prague 1, Czech Republic.
zitko@ms.mff.cuni.cz

obor studia:
M11 - vědecko-technické výpočty

Abstrakt

An important problem in optimization and linear algebra is the trust region problem: to minimize a quadratic function subject to an ellipsoidal or spherical constraint. There exist a lot of methods to solve this problem. Most of them lead to a system of linear equations and so may require matrix factorizations. This is expensive especially in the large scale optimization. In this paper it is shown that the solution of the trust region problem can be found by solving a parameterized eigenvalue problem.

1. Introduction

An important problem in optimization and linear algebra is the trust-region problem:

$$\min\{\psi(x), \ \ \| x \| \leq \Delta\}, \tag{7}$$

where

$$\psi(x) = \frac{1}{2}x^T A x + g^T x,$$

$$A \in \mathbb{R}^{n \times n}, \ \ A = A^T, \ \ x \in \mathbb{R}^n, \ \ g \in \mathbb{R}^n, \ \ \Delta > 0$$

and $\| \, . \, \|$ is the Euclidean norm.
It's well known that the point $x_\star \equiv x_\star(\Delta)$ is a solution to (7) if and only if

$$(A - \lambda_\star I)x_\star = -g \tag{8}$$

$$\text{with} \ \ \lambda_\star \leq 0, \ \ \lambda_\star(\Delta - \| x_\star \|) = 0$$

$$\text{and} \ \ A - \lambda_\star I \ \ \text{positive semidefinite.}$$

It follows from (8) that $g^T x_\star < 0$.

There are many different approaches to solve the basic problem (7):

1. The solution $x(\tilde{\Delta})$ can be found by an iterative method so that

$$\frac{|\tilde{\Delta} - \Delta|}{\Delta} \leq \varepsilon.$$

However, solving (8) for different λ may be expensive in the large scale optimization.

2. Steihaug [2] used the preconditioned conjugate gradient method to generate a piecewise linear approximation $\tilde{x}(\tau)$ that approximates the solution x_Δ. This is usually referred to as a dogleg or double dogleg technique. The aproximate curve is so that

   - $\psi(\tilde{x}(\tau))$ is monotonically decreasing
   - $\| \tilde{x}(\tau) \|$ is monotonically increasing for $0 \leq \tau \leq T$

   An aproximate solution to the problem (7) is found by solving

$$\min\{\psi(\tilde{x}(\tau)); \quad \| \tilde{x}(\tau) \| \leq \ \Delta, \quad 0 \leq \tau \leq T\}.$$

   In the dogleg methods the solution is easy to find as the search space is small, but the search space $\mathbb{R}^n$ in the problem (7) is large. This leads to the possibility of solving a compromise problem

$$\min\{\psi(x); \quad x \in S, \quad \| x \| \leq \Delta\}, \tag{9}$$

   where $S \subset \mathbb{R}^n$ is a specially chosen subspace.

   Considering $S \equiv K_{k+1} = span\{g, Ag, A^2g, ..., A^kg\}$, the Krylov space generated by the starting vector g and matrix A, the minimizer $x_\star$ of ψ is restricted to the $(k+1)$-dimensional subspace S.

   Now we can use two different techniques for generating the basis for the same Krylov space K_{k+1}.

   (a) The preconditioned conjugate gradient method [2] is very efficient for constructing the basis. It aims for an A-orthogonal basis $span\{p_0, p_1, ..., p_k\}$.

   (b) Gould [3] suggests another efficient method for generating the basis for K_{k+1}. It's the Lanczos method which obtains an orthonormal basis $span\{q_0, q_1, ..., q_k\}$.

   In both cases we seek $x_{k+1} \in S$, where x_{k+1} solves the problem (9).

3. If the matrix $A - \lambda I$ is positive definite then Moré and Sorensen [4] compute a Cholesky factorization

$$A - \lambda I = R_\lambda^T R_\lambda$$

   and apply Newton's method to find a solution to the equation

$$\Phi(\lambda) = 0, \quad \text{where} \quad \Phi(\lambda) = \frac{1}{\Delta} - \frac{1}{\| x_\lambda \|}$$

   This method is not appropriate for large scale problems which don't afford a Cholesky decomposition.

4. A very interesting method is developed by Sorensen [5, 6]. Defining a new parameter α, the problem (7) is replaced by a parameterized eigenvalue problem that is described below.

The paper is organized as follows. In the first two sections we analyze the structure of the problem, summarize our approach to the problem and present some details. In the fourth section we characterize the hard case of the parameterized eigenproblem. In the next sections we introduce safeguarding to assure global convergence of the iteration, stopping criteria for declaring convergence and show the superlinear rate of convergence. Simple numerical experiments are described in the last section.

2. Structure of the problem

Let's consider the basic problem

$$\min\{\psi(x), \ \ \| x \| \le \Delta\},$$

where

$$\psi(x) = \frac{1}{2}x^T A x + g^T x,$$

$$A \in \mathbb{R}^{n\times n}, \quad A = A^T, \quad x \in \mathbb{R}^n, \quad g \in \mathbb{R}^n, \quad \text{and} \quad \Delta > 0.$$

We define a new parameter α:

$$\frac{\alpha}{2} + \psi(x) = \frac{1}{2} \cdot (1, x^T) \cdot \begin{pmatrix} \alpha & g^T \\ g & A \end{pmatrix} \cdot \begin{pmatrix} 1 \\ x \end{pmatrix} \tag{10}$$

Let $y = (1, x^T)^T$, $\ e_1 = (1, 0, ...)^T \in \mathbb{R}^{n+1}$ and we define the bordered matrix

$$B_\alpha = \begin{pmatrix} \alpha & g^T \\ g & A \end{pmatrix}$$

Thus the problem (7) can be rewritten as

$$\min \ \frac{1}{2}y^T B_\alpha y, \quad y^T y \le 1 + \Delta^2, \quad e_1^T y = 1.$$

This formulation immediately leads to an eigenpair problem of the bordered matrix B_α: We seek x such that

$$\begin{pmatrix} \alpha & g^T \\ g & A \end{pmatrix} \cdot \begin{pmatrix} 1 \\ x \end{pmatrix} = \begin{pmatrix} 1 \\ x \end{pmatrix} \cdot \lambda$$

It follows from this that

$$(A - \lambda I)x = -g \tag{11}$$

$$\alpha - \lambda = -g^T x = g^T (A - \lambda I)^{-1} g = \sum_{j=1}^{n} \frac{\gamma_j^2}{\delta_j - \lambda}, \tag{12}$$

where $\{\delta_j\}$ are the eigenvalues of A and $\{\gamma_j\}$ are the expansion coefficients of g in the eigenvector basis. This means that δ_1 is the smallest eigenvalue of A and $\lambda_1(\alpha)$ is the smallest eigenvalue of B_α. Cauchy's interlace theorem says that the eigenvalues of B_α interlace the eigenvalues of A. Therefore

$$\lambda_1(\alpha) \le \delta_1 \quad \text{and} \quad A - \lambda_1(\alpha) I \quad \text{is positive semidefinite} \ \ \forall \alpha.$$

Let's define a function $\Phi(\lambda)$ which we'll need later.
<u>**Definition:**</u> Let

$$\Phi(\lambda) = g^T (A - \lambda I)^{-1} g = -g^T x,$$

and so

$$\Phi'(\lambda) = g^T (A - \lambda I)^{-2} g = x^T x.$$

Now we make a summary of our approach and show the meaning of the function $\Phi(\lambda)$.
<u>**Summary:**</u>

1. Finding the smallest eigenpair $\{\lambda, y\}$ of B_α for a given α.
2. Normalizing $y = (1, x^T)^T$.
3. Evaluation of the function Φ and its derivative at λ.
4. If α can be adjusted so the corresponding x satisfies $\Phi'(\lambda) = \Delta^2$ with $\alpha - \lambda = \Phi(\lambda)$, then
$$(A - \lambda I)x = -g \quad \text{and} \quad \lambda(\Delta - \| x \|) = 0$$
with $A - \lambda I$ positive semidefinite.
5. If $\lambda \le 0$, then x solves the trust-region problem.
6. If $\lambda > 0$ and $\| x \| \le \Delta$ will be found during the course of adjusting α, then A is positive definite and we can use the conjugate gradient method to the problem $Ax = -g$ and find the desired interior solution.

3. Interpolating scheme

We compute a function $\hat{\Phi}(\lambda)$ which interpolates Φ and Φ' at two points. From the interpolating function $\hat{\Phi}(\lambda)$ we determine $\hat{\lambda}$ satisfying $\hat{\Phi}'(\hat{\lambda}) = \Delta^2$. This $\hat{\lambda}$ and $\hat{\Phi}(\hat{\lambda})$ we use to update the parameter α satisfying $\alpha - \hat{\lambda} = \hat{\Phi}(\hat{\lambda})$ and compute the next point $\{\lambda, x\}$ from the updated bordered matrix B_α.

1. One-point method: Let

$$\hat{\Phi}(\lambda) = \frac{\gamma^2}{\delta - \lambda}$$

be the interpolation and assume that $\{\lambda_0, x_0\}$ corresponds to the initial α_0. It means that $\{\lambda_0, x_0\}$ is the smallest eigenpair of B_{α_0}. From (12) we have

$$\alpha - \lambda_0 = -g^T x_0 \quad \text{with} \quad (A - \lambda_0 I)x_0 = -g.$$

Now we require

$$\hat{\Phi}(\lambda_0) = \Phi(\lambda_0) = -g^T x_0 \quad \text{and} \quad \hat{\Phi}'(\lambda_0) = \Phi'(\lambda_0) = x_0^T x_0.$$

From these conditions one can compute coefficients γ^2 and δ:

$$\gamma^2 = \frac{(g^T x_0)^2}{x_0^T x_0} \quad \text{and} \quad \delta = \lambda_0 - \frac{g^T x_0}{x_0^T x_0} = \frac{x_0^T A x_0}{x_0^T x_0}$$

Finally we compute $\hat{\lambda}$ such that $\hat{\Phi}'(\hat{\lambda}) = \Delta^2$:

$$\hat{\lambda} = \delta + \frac{g^T x_0}{\| x_0 \| \cdot \Delta}$$

and update $\alpha_1 = \hat{\lambda} + \hat{\Phi}(\hat{\lambda})$.

2. When we have the first iterate we can continue with the two point method. This method is based on using the points

$$\lambda_{k-1}, \quad \lambda_k, \quad x_{k-1} \quad \text{and} \quad x_k$$

and values

$$\Phi(\lambda_{k-1}), \quad \Phi(\lambda_k), \quad \Phi'(\lambda_{k-1}), \quad \text{and} \quad \Phi'(\lambda_k).$$

Let's define

$$\hat{\Phi}(\lambda) = \frac{\gamma^2}{\delta - \lambda} + \eta \quad \text{for any} \quad \eta.$$

Firstly we compute $\hat{\lambda}$ such that $\frac{1}{\sqrt{\hat{\Phi}'(\hat{\lambda})}} = \frac{1}{\Delta}$ by applying interpolating philosophy:

$$\frac{1}{\Delta} = \frac{1}{\sqrt{\Phi'(\lambda_{k-1})}} \cdot \frac{\lambda_k - \hat{\lambda}}{\lambda_k - \lambda_{k-1}} + \frac{1}{\sqrt{\Phi'(\lambda_k)}} \cdot \frac{\hat{\lambda} - \lambda_{k-1}}{\lambda_k - \lambda_{k-1}} \tag{13}$$

Now we determine γ^2 and δ using (13) as above. For η we take $\Phi(\lambda_{k-1})$, $\Phi(\lambda_k)$ and $\hat{\lambda}$ and apply interpolating philosophy:

$$\eta = \frac{\lambda_k - \hat{\lambda}}{\lambda_k - \lambda_{k-1}} \cdot \eta_{k-1} + \frac{\hat{\lambda} - \lambda_{k-1}}{\lambda_k - \lambda_{k-1}} \cdot \eta_k,$$

where

$$\eta_j = \Phi(\lambda_j) - \frac{\gamma^2}{\delta - \lambda_j}, \quad j = k-1, \ k.$$

Finally we update

$$\alpha_{k+1} = \hat{\lambda} + \hat{\Phi}(\hat{\lambda}). \tag{14}$$

Now we summarize all our considerations into a simple algorithm.
The algorithm:

1. We seek α such that $\alpha - \lambda = \Phi(\lambda), \quad \Phi'(\lambda) = \Delta^2$,
   where $\Phi(\lambda) = -g^T x, \quad \Phi'(\lambda) = x^T x$.
2. Let $\{\lambda_{k-1}, x_{k-1}\}$ and $\{\lambda_k, x_k\}$ be the current iterations.
3. Let $\hat{\Phi}$ interpolates Φ at points $\{\lambda_{k-1}, x_{k-1}\}$ and $\{\lambda_k, x_k\}$.
4. We determine $\hat{\lambda}$ satisfying $\hat{\Phi}'(\hat{\lambda}) = \Delta^2$.
5. We update $\alpha_{k+1} = \hat{\lambda} + \hat{\Phi}(\hat{\lambda})$.
6. From $B_{\alpha_{k+1}}$ we obtain a new iteration $\{\lambda_{k+1}, x_{k+1}\}$.

4. The hard case

An unpleasant possibility which can occur is that the smallest eigenvector corresponding to the smallest eigenvalue of the bordered matrix B_α has its first component equal to zero and so cannot be orthogonalized to have it equal to one. This is equivalent to the so-called hard case. The iterates $\{\lambda_k, x_k\}$ then converge to $\{\delta_1, p\}$, where $(A - \delta_1 I)p = -g$.

Let $y = (\nu_1, u_1^T)^T$, where $|\nu_1| \leq \varepsilon$ and we define the space $S_1 = \{q; Aq = q\delta_1\}$ associated to δ_1.
Lemma: For each $\alpha \in \mathbb{R}$ and $q \in S_1$, $\{\delta_1, (0, q^T)^T\}$ is an eigenpair of B_α if and only if $g \perp S_1$.
Lemma: Suppose that $g \perp S_1$, $p = -(A - \delta_1 I)^+ g$, $\tilde{\alpha} = \delta_1 - g^T p$. Then $\{\delta_1, (1, p^T)^T\}$ is an eigenpair of $B_{\tilde{\alpha}}$. Moreover $(1, p^T)^T \perp (0, q^T)^T \quad \forall q \in S_1$. Here the symbol A^+ denotes the Moore-Penrose generalized inverse of A.

For all values of α greater than this critical value $\tilde{\alpha}$, the eigenvectors corresponding to the smallest eigenvalue of B_α will have a zero first component. When α exceeds $\tilde{\alpha}$, there is a well defined eigenvector that can be safely normalized. This parameterized vector corresponds to the second smallest eigenvalue of B_α.

How can be the hard case detected?
The vector $(\nu_1, u_1^T)^T$ is an eigenvector of B_{α_k} corresponding to the smallest eigenvalue $\lambda_1(\alpha_k)$. So

$$
\begin{array}{rcl}
\begin{pmatrix} \alpha_k & g^T \\ g & A \end{pmatrix} \cdot \begin{pmatrix} \nu_1 \\ u_1 \end{pmatrix} & = & \lambda_1(\alpha_k) \cdot \begin{pmatrix} \nu_1 \\ u_1 \end{pmatrix} \quad \Leftrightarrow \\
\Leftrightarrow \quad (A - \lambda_1(\alpha_k) \cdot I) \cdot u_1 & = & -g \cdot \nu_1 \quad \Leftrightarrow \\
\Leftrightarrow \quad \dfrac{\| (A - \lambda_1(\alpha_k) \cdot I) \cdot u_1 \|}{\| u_1 \|} & = & \dfrac{\| g \| \cdot |\nu_1|}{\sqrt{1 - \nu_1^2}}
\end{array}
$$

Hence $\| g \| \cdot |\nu_1| \leq \varepsilon \cdot \sqrt{1 - \nu_1^2}$ assures that

$$\| (A - \lambda_1(\alpha_k) \cdot I) \cdot u_1 \| \leq \varepsilon \cdot \| u_1 \|$$

and we have $\{\lambda_1(\alpha_k), u_1\}$ an aproximate eigenpair of A.

For defining the point $\{\lambda_k, x_k\}$ at each iteration we compute the two smallest eigenpairs of B_{α_k} :

$$\{\lambda_1(\alpha_k), (\nu_1, u_1^T)^T\} \quad \text{and} \quad \{\lambda_2(\alpha_k), (\nu_2, u_2^T)^T\}$$

and whenever the hard case is detected, we simply use the second smallest eigenpair of B_{α_k} :
If $|\nu_1|$ is too small, that is, if $\| g \| \cdot |\nu_1| \leq \varepsilon \cdot \sqrt{1 - \nu_1^2}$,
then we set $\lambda_k = \lambda_2(\alpha_k)$ and $x_k = \frac{u_2}{\nu_2}$,
otherwise $\lambda_k = \lambda_1(\alpha_k)$ and $x_k = \frac{u_1}{\nu_1}$.

5. Safeguarding

Safeguarding is important to assure global convergence of the iteration. For the optimal parameter $\alpha_\star = \lambda_\star - g^T x_\star$ an enequality

$$\delta_1 - \frac{\| g \|}{\Delta} \;\le\; \alpha_\star \;\le\; \delta_1 + \| g \| \cdot \Delta \tag{15}$$

holds. However, computing δ_1 or a good approximation to δ_1 can be expensive, so we replace it by a suitable bound. Let $\delta_S = \min\{a_{ii};\ i = 1, ..., n\}$, alternatively if the diagonal of A isn't available, $\delta_S = \frac{v^T A v}{v^T v} \;\ge\; \delta_1$ with v randomly chosen.
Thus we set

$$\alpha_U \equiv \delta_S + \| g \| \cdot \Delta$$

as the upper bound for the optimal $\alpha_\star$. Moreover, $\alpha \le 0$ implies B_α is not positive definite, so we set $\alpha_0 = \min\{0, \alpha_U\}$ to assure that $\delta_I \equiv \lambda_1(\alpha_0) \le 0$. Now we define the lower bound

$$\alpha_L \equiv \delta_I - \frac{\| g \|}{\Delta},$$

since the interlacing property implies $\delta_I \le \delta_1$. Thus we have

$$\alpha_L \le \alpha_\star \le \alpha_U. \tag{16}$$

From (12) and (16) we obtain initial bounds for the sequence in λ:

$$\lambda_L \equiv \delta_I - \| g \| \cdot \left(\frac{1}{\Delta} + \Delta \right) \;\le\; \lambda_\star \;\le\; \lambda_U \equiv\; \delta_S.$$

The point δ_S is updated every iteration:

$$\delta_S = \min\{\delta_S, \frac{u_1^T A u_1}{u_1^T u_1}\}, \quad \text{where} \quad \frac{u_1^T A u_1}{u_1^T u_1} = \lambda_1(\alpha_k) - \nu_1 \cdot \frac{g^T u_1}{u_1^T u_1}$$

Whenever α_{k+1} computed by (14) doesn't belong to $< \alpha_L, \alpha_U >$, we use

$$\alpha_{k+1} = \frac{\alpha_L + \alpha_U}{2}$$

instead. In both cases we have

$$\lambda_1(\alpha_{k+1}) \in < \lambda_L, \lambda_U > .$$

Let $(\nu_1, u_1^T)^T$ be an eigenvector of $B_{\alpha_{k+1}}$ corresponding to $\lambda_1(\alpha_{k+1})$. If $\| x_{k+1} \| < \Delta$, we update

$$\lambda_L = \lambda_1(\alpha_{k+1}) \;\text{ and }\; \alpha_L = \alpha_{k+1},$$

otherwise

$$\lambda_U = \lambda_1(\alpha_{k+1}) \;\text{ and }\; \alpha_U = \alpha_{k+1}.$$

In other words, the lengths of $< \alpha_L, \alpha_U >$ and $< \lambda_L, \lambda_U >$ are reduced at every iteration.

6. Stopping criteria

We include the following result which is useful for the hard case.
Lemma: Let $\varepsilon \in (0, 1)$ be given and suppose

$$(A - \lambda I)p = -g, \quad \lambda \le 0 \;\text{ with }\; A - \lambda I \;\text{ positive semidefinite.}$$

If $\| p + z \| = \Delta$ and $z^T (A - \lambda I) z \le -\varepsilon (g^T p + \lambda \Delta^2)$, then

$$\psi_\star \le \psi(p + z) \le \frac{1}{2}(1 - \varepsilon)(g^T p + \lambda \Delta^2) \le (1 - \varepsilon)\psi_\star,$$

where $\psi_\star \le 0$ is the optimal value of (7).
Now the convergence of iterates is stated as follows.
Convergence:

1. If $\Big| \| x_k \| - \Delta \Big| \le \varepsilon_\Delta \cdot \Delta$, then x_k is a binding solution of the trust region problem.

2. If the hard case occurs, that is, if $\| g \| \cdot |\nu_1| \le \varepsilon_\nu \cdot \sqrt{1-\nu_1^2}$, then:
   If $\| x_k \| < \Delta$ and $\lambda_k \le \delta_S$ then
   - $z = \frac{u_1}{\|u_1\|}$ is an aproximation to an eigenvector of A corresponding to δ_1
   - we compute τ such that $\| x_k + \tau \cdot z \| = \Delta$
   - if $\tau^2 \cdot (z^T A z - \lambda_k) \le -\varepsilon_{HC} \cdot (g^T x_k + \lambda_k \Delta^2)$ then $x_k + \tau \cdot z$ is a solution.

3. If $\| x_k \| < \Delta$ and $0 < \lambda_k \le \delta_S$ then the solution to (7) is inside the trust region and the matrix A is positive definite. So we use conjugate gradient method to the linear system $Ax = -g$.

7. Local convergence analysis

The next results give a superlinear rate of convergence.
Lemma: Let $\{\lambda_\star, x_\star\}$ be a solution to the problem (7) with $\| x_\star \| = \Delta$. Then there is a neighbourhood $\mathcal{B}$ of $\lambda_\star$ such that if $\lambda_{k-1}, \lambda_k \in \mathcal{B}$ then the iterate λ_{k+1} satisfies

$$\lambda_{k+1} - \lambda_\star \le C \cdot (\lambda_{k-1} - \lambda_\star)(\lambda_k - \lambda_\star).$$

Theorem: Assume that the problem (7) has a solution $\{\lambda_\star, x_\star\}$ with $x_\star$ on the boundary of the trust region. Then there exists a neighbourhood $\mathcal{B}$ of $\lambda_\star$ such that the sequence of iterates $\{\lambda_k, x_k\}$ using the two-point scheme beginning with $\lambda_0, \lambda_1 \in \mathcal{B}$ is well defined, remains in $\mathcal{B}$ and converges superlinearly to $\lambda_\star$. Moreover, if $\lambda_\star < \delta_1$, the sequence $\{x_k\}$ converges superlinearly to $x_\star$ and, if $\lambda_\star = \delta_1$, then $\{x_k\}$ converges superlinearly to a vector p such that $(A - \delta I)p = -g$, $\| p \| < \Delta$.

8. Numerical experiments

In this last section we present some simple numerical experiments to demonstrate the behaviour of the algorithm. The matrix A on problem (7) was 3-diagonal of orders 100 and 400. The diagonals were introduced to make the matrix indefinite. The vector g was randomly generated with entries from $(-0.5; 0.5)$. The trust region radius was 10^{-3}, 10^{-1} and 10^2. The problems were solved with the tolerances $\varepsilon_\Delta = 10^{-6}$ and 10^{-8}. Our results are summarized in the table. The average number of iterations for detecting the approximate solution is in the fourth column. In all cases the solution lies on the boundary of the trust region. In the last column there is the average error of computed solution from the trust region radius.

ε_Δ	N	Δ	*Iterations*	*Error*	ε_Δ	N	Δ	*Iterations*	*Error*
10^{-6}	100	10^{-3}	3,5	$3,3 \cdot 10^{-9}$	10^{-8}	100	10^{-3}	3,5	$8,1 \cdot 10^{-11}$
		10^{-1}	4	$1,8 \cdot 10^{-9}$			10^{-1}	4,5	$1,2 \cdot 10^{-10}$
		10^{2}	3	$4,8 \cdot 10^{-6}$			10^{2}	5	$2,7 \cdot 10^{-9}$
	400	10^{-3}	3,5	$7,8 \cdot 10^{-9}$		400	10^{-3}	4	$3,3 \cdot 10^{-13}$
		10^{-1}	4	$7,8 \cdot 10^{-10}$			10^{-1}	4,5	$3,6 \cdot 10^{-10}$
		10^{2}	4,5	$3,5 \cdot 10^{-5}$			10^{2}	6	$9,9 \cdot 10^{-10}$

9. Conclusion

A few methods have been developed for solving the trust region problems. We mainly aimed at the possibility to find the solution of the trust region problem by solving a parameterized eigenvalue problem of a bordered matrix. Numerical experiments we have performed show that this method seems to be effective and robust for a special class of problems. Further experience with testing the more complicated examples and especially the examples where the hard case occurs will be necessary.

References

[1] L. Lukšan: Numerické optimalizační metody pro úlohy bez omezujících podmínek (květen 1995)

[2] T. Steihaug: The conjugate gradient method and trust regions in large scale optimization (October 1981)

[3] N. Gould, S. Lucidi, M. Roma, P. Toint: Solving the trust region subproblem using the Lanczos method (June 1997)

[4] J. Moré, D. Sorensen: Computing a trust region step (March 1981)

[5] D. Sorensen: Minimization of a large scale quadratic function subject to a spherical constraint (September 1994)

[6] S. Santos, D. Sorensen: A new matrix free algorithm for the large scale trust region subproblem (June 1995)

H-verze kontaktního problému v termopružnosti

doktorand:
MGR. ZDENĚK KESTŘÁNEK
ÚI AV ČR, Pod vodárenskou věží 2, 182 07 Praha 8
zdenda@uivt.cas.cz

školitel:
DOC. ING. JIŘÍ NEDOMA, CSC.
ÚI AV ČR, Pod vodárenskou věží 2, 182 07 = Praha 8
nedoma@uivt.cas.cz

obor studia:
Matematické modelování

Abstrakt

V příspěvku jsou diskutovány aspekty týkající = se adaptivního zjemňování sítě konečných prvků pro = prostorové kontaktní úlohy. Pro zjemnění je užito strategie půlení = čtyřstěnů. Tato strategie produkuje sítě dobré kvality a je = relativně snadno analyzovatelná a implementovatelná. Algoritmus bylo třeba = vhodným způsobem upravit na kontaktní části hranice. Elementy sítě jsou nejdříve zjemněny na základě a-posteriorního = odhadu a poté je provedeno zjemňování další, sloužící k = zachování konformity sítě. Ukážeme několik a-posteriorních estimátorů, = které jsou založeny na principu zhlazení gradientu a na = měření skoků napětí mezi jednotlivými elementy.

1. Úvod

Adaptivní zjemňování sítě se ukázalo být = efektivním nástrojem při numerickém řešení parciálních = diferenciálních rovnic. Snížení diskretizační chyby je dosaženo s co = nejmenším počtem dodatečných stupňů volnosti. Poznamenejme, že zjemňování sítě představuje = h-verzi metody konečných prvků, zatímco v p-verzi se zvyšuje řád aproximačních = funkcí. Ke zjemnění sítě dochází pouze v oblastech, kde = odhadnutá chyba přibližného řešení je veliká. V mechanice = pružnosti se může jednat o oblasti v blízkosti uchycení, = kontaktu, případně v blízkosti nekonvexních úhlů.

Proces adaptivního zjemňování sítě má = následující obecnou strukturu:

(1) Sestrojení počáteční hrubé = sítě $\mathcal{T}^0$; $k = 3D0$

(2) Řešení problému na $\mathcal{T}^k$

(3) Vypočtení odhadu chyby estimátorem pro = všechny elementy $\mathcal{T}^k$ a vytvoření $\mathcal{S}^k \subset = \mathcal{T}^k$, kde $\mathcal{S}^k$ je množina elementů s velkým odhadem chyby

(4) Pokud $\mathcal{S}^k \neq \{\emptyset\}$,
    dělení elementů z $\mathcal{S}^k$;
    vytvoření nové konformní sítě = $\mathcal{T}^{k+1}$; $k = 3Dk + 1$

návrat na (2)

Jinak

konec

Kroky (2)-(4) pro pevné k nazveme iterací zjemnění. Iterace v (4) vedoucí ke konformní síti v k-té = iteraci zjemnění nazveme iteracemi konformity. Pro kroky (3) a (4) je třeba volit takové datové struktury a algoritmy, které lze relativně snadno zařadit do = existujících programů pro řešení kroku (1) a zejména kroku (2). Přirozeně je nutno brát v = úvahu také otázky robustnosti a časové a paměťové = efektivity.

Mezi první práce zabývající se problémem = estimátorů se považují články [2] a [3]. Ve [3] je analyzován v jedné dimenzi = estimátor využívající vlastních čísel příslušného operátoru. [2] je založen na řešení lokálních = Dirichletových problémů, pro každý takový lokální problém je ovšem třeba = uvažovat několik elementů současně. Autoři [8] odvodili estimátor, kde jsou brány v úvahu jak napěťové skoky na hranicích elementů, tak = přímo rezidua odpovídající rovnice. V případě lineární aproximace je reziduální = člen nulový. Získáváme tak estimátor podobný [4]. V = příspěvku porovnáváme vlastnosti estimátorů [12], [4] a [5]. [12] je založen na principu zhlazení gradientu, [4] představuje residuální typ estimátoru a [5] je založen na řešení lokálních = Neumannových problémů.

Další část kroku (3) určuje elementy s velkým = odhadem chyby. existují dvě základní strategie. V první = požadujeme, aby celková chyba, tj. součet chyb na jednotlivých elementech, byla menší než předepsaná tolerance ε. = Postačující podmínka přijatelnosti sítě s N elementy je

$$\eta_T \leq \frac{\varepsilon}{\sqrt{N}} \qquad \forall T \in T_k. \tag{17}$$

Děleny jsou elementy, kde

$$\eta_T > \frac{\varepsilon}{\sqrt{N}}. \tag{18}$$

Ve druhé strategii (fixed fraction) jsou elementy seřazeny dle odhadnutých chyb a k dělení je určena určitá jejich = část s největší hodnotou chyby

$$\eta_T > \gamma \cdot \max_{T' \in T_k} \eta_{T'}, \qquad 0 < \gamma < 1. \tag{19}$$

Podobně jako ve dvou dimenzích, může být = čtyřstěn dělen buď oktasekcí nebo bisekcí. V dalším se zaměřujeme na bisekci. První = algoritmus tohoto typu byl publikován v [6] a je v podstatě ekvivalentní algoritmu [1]. Jiný přístup je uveden v [10], který = vychází ze zajištění konformity pro stěny jednotlivých = elementů. Algoritmus implementovaný v této práci vychází z [1] a obsahuje = potřebné modifikace pro zajištění konformity na kontaktních = hranicích.

2. Formulace kontaktního problému

Nechť $\Omega \subset R^3$ je oblast s hranicí $\partial\Omega$. Nechť se tato oblast skládá z S podoblastí $\Omega = 3D \bigcup_{\iota=3D1}^{S} \Omega^\iota$. Nechť $\partial\Omega = 3D\Gamma_u \cup \Gamma_\tau \cup = \Gamma_c^{kl} \cup \Gamma_0$, $1 \leq k < l \leq S$.

Uvažujme lineární termopružnost určenou Hookovým = zákonem

$$\tau_{ij} = 3Dc_{ijkl}e_{kl} - \beta_{ij}(T - T_0), \quad i,j,k,l = 3D1,2,3, \tag{20}$$

kde τ_{ij} je tenzor napětí, c_{ijkl} jsou symetrické koeficienty elasticity, e_{ij} je tenzor deformace

$$e_{ij} = 3De_{ij}(\mathbf{u}) = 3D\frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \tag{21}$$

β_{ij} jsou koeficienty tepelné roztažnosti. T, resp. = T_0 je teplota, resp. teplota beznapěťového stavu. Tenzor napětí splňuje rovnici rovnováhy

$$\frac{\partial \tau_{ij}}{\partial x_j} + f_i = 3D0, \tag{22}$$

kde f_i jsou složky vektoru vnějších sil. Na $\partial\Omega$ definujme

$$u_n = 3Du_i n_i, \qquad u_{tj} = 3Du_j - u_n n_j, \tag{23}$$

$$\tau_n = 3D\tau_{ij} n_i n_j, \qquad \tau_{tj} = 3D\tau_{jk} n_k - \tau_n n_j, \tag{24}$$

kde n_i jsou složky vnější normály k $\partial\Omega$. Okrajové podmínky jsou následujícího tvaru

$$u_i = 3Du_{0i} \qquad \text{na } \Gamma_u, \tag{25}$$

$$\tau_{ij} n_j = 3DP_i \qquad \text{na } \Gamma_\tau, \tag{26}$$

$$u_n^k + u_n^l \leq 0, \quad \tau_n^k = 3D\tau_n^l \leq 0, \quad (u_n^k + u_n^l)\tau_n^k = 3D0 \qquad \text{na } \Gamma_c^{kl}, \tag{27}$$

$$\tau_t^k = 3D - \tau_t^l, \quad |\tau_t^k| \leq g_c^{kl} \equiv \mathcal{F}_c^{kl} |\tau_n^k| \qquad \text{na } \Gamma_c^{kl}, \tag{28}$$

$$|\tau_t^k| < g_c^{kl} \quad \Rightarrow \quad |\mathbf{u}_t^k - = \mathbf{u}_t^l| = 3D0 \qquad \text{na } \Gamma_c^{kl}, \tag{29}$$

$$|\tau_t^k| = 3Dg_c^{kl} \quad \Rightarrow \quad \exists\lambda \geq= 0, \quad \tau_t^k = 3D - \lambda(\mathbf{u}_t^k - \mathbf{u}_t^l) \qquad \text{na } = \Gamma_c^{kl}, \tag{30}$$

$$u_n = 3D0, \quad \tau_{tj} = 3D0 \qquad \text{na } \Gamma_0. \tag{31}$$

Zde $\mathbf{u}_0$ je vektor předepsaného posunutí, P_i jsou složky vektoru zadaného napětí, (27) jsou Signoriniho kontaktní podmínky, podmínky (28-30) vyjadřují = Coulombův zákon tření. Podmínka (31) je oboustranný kontakt. Hledanou funkcí je pole posunutí $\mathbf{u}$.

Odpovídající variační formulace je = aproximována MKP [9], kde uvažujeme konformní síť, tj. dva = čtyřstěny sdílejí buď celou stěnu, celou hranu nebo vrchol nebo jejich průnik je = prázdný. Přicházíme tak k diskretizované verzi problému = sedlového bodu [9].

3. Odhad chyby

U všech estimátorů vycházíme z lineárních = problémů.

Prvním je Z^2 estimator [12]. Je měřena L^2 norma $\tau^* - \tau_h$, kde τ_h je vypočtená aproximace τ a τ^* je nějaká lepší aproximace τ než τ_h.

$$\eta_{T,1} \equiv ||\tau^* - \tau_h||_{L^2(T)} \tag{32}$$

Řád této lepší aproximace je o jeden stupeň = vyšší než τ_h, tj. τ^* je po částech lineární.

Pro druhý estimátor [4] zavedeme několik označení. Uvažujme dvojici čtyřstěnů T_k, T_m, $k < m$,, které sdílejí společnou stěnu. Nechť $\Gamma_I = 3D \bigcup\limits_{k<m} T_k \cap T_m$. Nechť $[[\tau]] = 3D\tau|_{T_m} - \tau|_{T_k}$, $\mathbf{n} = 3D(n_i)_{i=3D1,2,3}$ je vnější normála k T_k. Definujme lokální projekce vstupních dat

$$\Pi_T \mathbf{F} = 3D\frac{\int_T \mathbf{F} d\mathbf{x}}{meas\ T} \qquad \text{kde } meas\ T \text{ je objem čtyřstěnu}, \tag{33}$$

$$\Pi_S \mathbf{P} = 3D\frac{\int_S \mathbf{P} d\mathbf{s}}{meas\ S} \qquad \text{kde } meas\ S \text{ je plocha stěny } \Gamma_\tau. \tag{34}$$

Nechť konečně

$$\mathbf{J}_s = 3D \begin{cases} [[\tau_{ij}(\mathbf{u}_h) n_j]]_S & S \subset \Gamma_I \\ 2\{\Pi_S \mathbf{P} - (\tau_{ij}(\mathbf{u}_h) n_j)|_S\} & S \subset \Gamma_\tau \\ 0 & S \subset \Gamma_u \end{cases} \tag{35}$$

Druhý estimátor je

$$\eta_{T,2} \equiv \left[(meas\ T)^2 |\Pi_T \mathbf{F}|^2 + \frac{1}{2} \sum_{S \in E_T} (meas\ S)^2 |\mathbf{J}_S|^2 \right]^{\frac{1}{2}}, \tag{36}$$

kde $E_T = 3DT \cap (\Gamma_\tau \cup \Gamma_u \cup \Gamma_I)$. Jelikož jsou známy uzlové hodnoty napětí, není = nutno provádět hledání sousedů daného elementu.

Je zřejmé, že $\eta_{T,2}$ explicitně závisí na = okrajových podmínkách, které zahrnují duální veličiny, tj. = napětí. Je měřeno reziduum mezi předepsanými a = počítanými hodnotami. Vliv kontaktní hranice bude tedy vyjádřen těmi = podmínkami v (27-30), které obsahují pouze duální = veličiny. U nerovnostních podmínek budou měřeny hodnoty $[\tau_n]^+$ a $[|\tau_t| - g_c]^+$. Podmínky akce a reakce vedou opět na měření skoků napětí. podmínky příliš

Poslední estimátor $\eta_{T,3}$ bude řešením = lokálních problémů s pravou stranou obsahující členy $\mathbf{J}_S$ (35) v prostoru "bublinových" funkcí. Tento estimátor je analogií estimátoru [5]. Jako prostor bublinových funkcí je zvolen prostor polynomů po částech kvadratických s nulami ve vrcholech čtyřstěnů. Pro každý = element je nutno sestavit a řešit systémy až 18 rovnic.

4. Algoritmus lokálního zjemnění

Vraťme se nyní ke kroku (4) obecného algoritmu = úvodní části. Arnoldův algoritmus lokálního zjemnění [1] = dosahuje konformity sítě iterativní bisekcí těch čtyřstěnů, = které podmínku konformity nesplňují. Důležitým krokem tedy je výběr = tzv. dělené hrany pro každý čtyřstěn sítě. Navíc je na každé stěně čtyřstěnu zvolena = tzv. označená hrana. Je vyžadováno, aby dva čtyřstěny, které = sdílejí společnou stěnu, měly pro tuto stěnu zvolenou stejnou hranu jako označenou. Je třeba vhodně označit hrany v počáteční = síti, přitom již bereme v úvahu dvojice kontaktních stěn. U následníků děleného čtyřstěnu = algoritmus určuje nové označené a dělené hrany.

5. Implementace algoritmu zjemnění

Data geometrie modelu jsou uložena v integerovém a realovém polích. V poli $ITNODE(J, I)$, $1 \leq I \leq NT$, kde NT je počet = elementů, jsou pro $1 \leq J \leq 4$ uloženy indexy vrcholů, a pro $J = 3D5$ index materiálu. Obdobně pro pole $IBNDRY(J, I)$, $1 \leq I \leq NB$, $1 \leq J \leq 5$, kde NB je počet hraničních stěn, $ICP(J, I)$, $1 \leq I \leq= NCP$, $1 \leq J \leq 2$, kde NCP je počet dvojic kontaktních = uzlů, $ICCB(J, I)$, $1 \leq I \leq NCB$, $1 \leq J \leq 3$, kde NCB je počet dvojic kontaktních = stěn. Informace ohledně oboustranného kontaktu je uložena = obdobně. Realová pole obsahují souřadnice vrcholů, = směrů normál, atd.

Pro adaptivní zjemňování je výhodné zavést pole = hran $IEDGE(J, I)$, $1 \leq I \leq NED$, $1 \leq J \leq 3$, kde NED je počet hran sítě. Toto pole je uspořádáno lexikograficky. Kromě indexů vrcholů je uchována informace o = následnících dané hrany, $IE3 = 3DIEDGE(3, I)$. Pokud $IE3 = 3D0$, nemá hrana I následníka. Pokud $IE3 < 0$, byl následník = vytvořen v právě probíhající iteraci konformity. Pokud = $IE3 > 0$, potom $IE3$ je ukazatel do pole $IEDGE$ na místo, kde je = uložena první následná hrana. Navíc přidáváme šestou složku do pole $ITNODE$, = která obsahuje informaci o dělené a označených hranách, $IETYPE = 3DITNODE(6, I)$. Pro každý čtyřstěn platí

$$1122 \leq |IETYPE| \leq 6655 \,. \tag{37}$$

REFERENCES

[1] D.N. Arnold, A. Mukherjee and L. Pouly, Locally adapted tetrahedral meshes using bisection, submitted to *SIAM J.Sci.Comp.* (1999).

[2] I. Babuška, W.C. Rheinboldt, Error estimates for adaptive finite element computations, *SIAM J.Num.Anal.* 15 (1978) 736-754.

[3] I. Babuška, W.C. Rheinboldt, A-posteriori error estimates for = the finite element method, *Int.J.Num.Meth.Eng.* 12 (1978) 1597-1615.

[4] I. Babuška, R. Durán and R. Rodríguez, Analysis of the efficiency of an a posteriori error estimator for linear triangular finite elements, *SIAM J.Num.Anal.* 29 (1992) 947-964.

[5] R.E. Bank and A. Weiser, Some a posteriori error estimators for elliptic partial differential equations, *Math.Comp.* 44 (1985) 283-301.

[6] E. Bänsch, Local mesh refinement in 2 and 3 dimensions, *Imp.Comp.Sci.Eng.* 3 (1991) 181-191.

[7] R. Becker, R. Rannacher, *A feed-back approach to error control in finite element methods: Basic analysis and examples* (Technical Report 359, Universität Heidelberg, 1996) 27p.

[8] C. Johnson and P. Hansbo, Adaptive finite element methods in computational mechanics, *Comp.Meth.Appl.Mech.Eng.* 101 (1992) 143-181.

[9] Z. Kestřánek, J. Nedoma, *FEC - A code for contact problems in thermoelasticity with friction* (Technical Report V-740, ICS AS CR, Prague, 1998) 12p.

[10] A. Plaza, M.A. Padrón, G.F. Carey, A 3D refinement/derefinement algorithm for solving evolution problems. In: Achim Sydow, Ed., *Proc.15th IMACS World Congress on = Scientif.Comp., Modell. and Appl.Math. Vol.2 Num.Math.* (Berlin, 1997) 197-202.

[11] R. Verfürth, A review of a-posteriori error estimation and mesh-refinement techniques, *J.Comp.Appl.Math.* 50 (1994) 67-83.

[12] O.C. Zienkiewicz, J.Z. Zhu, A simple error estimator and adaptive procedure for practical engineering analysis, *Int.J.Num.Meth.Eng.* 24 (1987) 337-357.

Buněčná síť - model počítače s proměnlivou architekturou

doktorand:
PŘEMYSL ŽÁK
ÚI AV ČR, Pod vodárenskou věží 2, 182 07 Praha 8

zak@uivt.cas.cz

školitel:
MARCEL JIŘINA ST.
ÚI AV ČR

marcel@uivt.cas.cz

obor studia:
Výpočetní technika - Neuronové sítě

Abstrakt

Předmětem tohoto příspěvku je nový model počítače inspirovaný poměry v imunitním systému. Základní výkonný prvek systému - buňka vyměňuje s ostatními signál podobně jako neuron v neuronové síti. Buňka má však navíc schopnost pohybu, takže se její partneři v komunikaci mění. Navíc se buňka může i rozdělit, nebo zaniknout.

1. Úvod

Jednou z vlastností většiny výpočetních systémů je pevná architektura. Je to dáno historickým vývojem a svázáno s možností praktické realizace. Svět kolem nás je však proměnlivý proto i informační systémy v přírodě mají často proměnlivou strukturu. Příkladem takového systému je třeba imunitní systém živých organismů [2, 3].

Imunitní systém, je komplex složitých reakcí ruzných molekul a buněk. Jeho popis by byl nejméně tak komplikovaný jako popis

systému nervového. Úkol obou systémů je stejný. Zpracovat signály okolního světa tak, aby organismus mohl přežít. Mnoho mechanismů zpracování informace v obou systémech je podobných, základní rozdíl je však právě v architektuře.

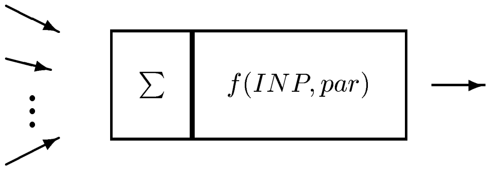
Co však přináší proměnlivá struktura a jaká jsou její úskalí?

V této práci je navržen počítačový model, který ke klasické neuronové síti přidává prvek proměnlivé architektury a proměnlivého počtu neuronů (zde buněk). Vzniká tak systém s úplně jiným stylem zpracování signálu. Vstup je rozprostřen náhodně mezi buňky a výstupem mohou být parametry systému, nebo přímo jeho reakce. Stabilní stav systému není vyjádřen konstantní hodnotou jistých parametrů, ale jejich kmitáním v omezených mezích.

2. Základní vymezení

Buňka - Objekt se svými parametry a funkcemi. Jedním z mnoha parametrů buňky je její poloha.

přijímá signál ze svého blízkého okolí, zpracuje ho a vyhodnotí jeho velikost. Výsledek jednak pošle do svého okolí a zároveň ho použije pro sebe.



Obrázek 5: Schéma buňky

V síti se pohybuje několik typů buněk. Každý typ má své specifické hodnoty některých parametrů.

Síť - obdélník m x n možných pozic buněk. Buňky v každém kroku skončí na určitém místě. Na jednom místě však může být najednou i více buněk.

Signál - Každá buňka v každém kroku vyšle signál na základě vstupního signálu z minulého kroku a zároveň přijme vstupní signál nový. Vstupní signál je poscítán ze signálů buněk z nejbližšího okolí za použití vzdálenosti, vah a prahu. Na tento signál se pak aplikuje aktivační funkce buňky, kterou je jistá zvonovitá funkce. Na základě spočítané aktivity a se buňka dělí nebo zaniká. Aktivita buňky je pak vyjádřena v dalším kroku jako její výstupní signál.

Motivace - využití vlastností živých systémů k obohacení repertoáru neuronových sítí.

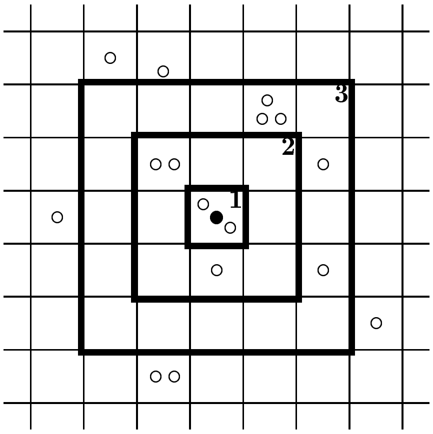
Cíl - Vytvořit stabilní systém se schopností učení.

3. Popis Buněčné sítě

Stavba sítě:

Celá síť je tvořena plochou (třeba čtvercem) skládající se z malých plošek (čtverečků), které představují místa výskytu buněk. Počet buněk přítomných najednou každém takovém místě je omezen.

Buňky se pohybují v jednotlivých krocích po síti. K každém kroku je prozkoumáno okolí buňky, z kterého může buňka přijmout signál.



Obrázek 6: Zóny působení buňky

V daném kroku buňka nejprve zpracuje signál pomocí vztahu

$$INP_i = \sum_{j=1}^{S} \frac{w_{i,j}}{l_{i,j}} X_{i,j} - THR_i \quad (38)$$

INP_i	*vstup i-té buňky*
S	*počet buněk v okolí*
$w_{i,j}$	*váha vztahu mezi i-tou a j-tou buňkou*
$l_{i,j}$	*vzdálenost mezi i-tou a j-tou buňkou*
$X_{i,j}$	*signál přicházející od j-té do i-té buňky*
THR_i	*práh i-té buňky*

Dále pak použije svou aktivační funkci. V praxi se nejlépe osvědčila zvonovitá funkce vzniklá odečtením dvou sigmoid, která svým tvarem nejlépe vystihuje biologickou skutečnost.

$$DSigm(INP, posunuti) = Sigm(INP) - Sigm(INP - posunuti) \quad (39)$$

Parametry sítě:

PRO CELOU SÍŤ
I x J - rozměry sítě
K - maximální počet buněk na jednom místě
N - počet typů buněk
seed number - pro generátor pseudonáhodných čísel
temperature - maximální vzdálenost přesunu buňky
ampl - zesilovací konstanta signálu v síti

PRO KAŽDÝ TYP ZVLÁŠŤ
name - název typu
max, steep - parametry aktivační funkce
alpha - koeficient rychlosti změn při učení
per - perioda působení učícího signálu
A - maximální věk buněk
supp - frekvence přísunu nových buněk
ND - perioda přirozeného dělení buněk
[0] - inicializační intervaly hodnot parametrů
W_i - váhový vektor

PRO JEDNOTLIVÉ BUŇKY
ID - identifikační číslo buňky
age - věk
i, j - pozice buňky

PARAMETRY UČENÍ (možné)
rec - práh buňky
mort, div - mez zániku a rozdělení buňky
lrn_i - individuální váhový vektor

Činnost sítě:

Nejprve se síť inicializuje. Nastavují se počáteční hodnoty parametrů platných pro celý model (seed, rychlost pohybu, parametry učení …), platných pro jeden typ buněk (počet buněk v typu, limit pro dělení a zánik buněk, váhy …) i pro jednotlivé buňky (identifikační číslo, poloha, věk …).

Dalším krokem je příjem signálu z okolí buňky a jeho zpracování popsané výše.

Výsledek tohoto zpracování se pak porovnává s mezemi pro dělení a zánik. Je li výsledná aktivita příliš malá, buňka zaniká, je-li naopak dostatečně vysoká, buňka se rozdělí. Překročí-li buňka svůj maximální věk, rovněž zaniká. Buňka se zde může též rozdělit přirozeným dělením, které probíhá náhodně s předem určenou četností.

Nakonec se buňky přemístí na své nové místo. Přesun je náhodný maximální vzdálenost přesunu je jedním z parametrů.

4. Diskuse

V předchozí kapitole je popsán poměrně komplikovaný systém. Bude proto dobré se na něj podívat z různých pohledů a zařadit jej mezi existující modely.

Cyklická neuronová síť

Představme si plně propojenou cyklickou síť, kde každý neuron představuje jeden typ buněk buněčné sítě. Komunikaci Buněčné sítě je tedy možno vzít jako komunikaci celých skupin buněk (typů) a vyjádřit ji cyklickou sítí.

Co však představuje základní parametry této cyklické sítě? Práh každého neuronu si můžeme představit jako nějakou funkci prahů buněk, osažených v příslušném typu. Váha pak souvisí s váhami typu. Velikost přeneseného signálu je však složitou funkcí jak aktivit buněk typu, tak jejich množství i počtu buněk v typu signál přijímající. Zde tato paralela přestává poněkud fungovat.

Kohonenova mapa

První pohled na obrázek buněčné sítě připomíná Kohonenovu mapu. Dalo by se však říct, že je to skoro vše. Dokonce i rostoucí síť s proměnlivým počtem neuronů [4] připomíná náš model velmi vzdáleně.

Konglomeráty - agregáty

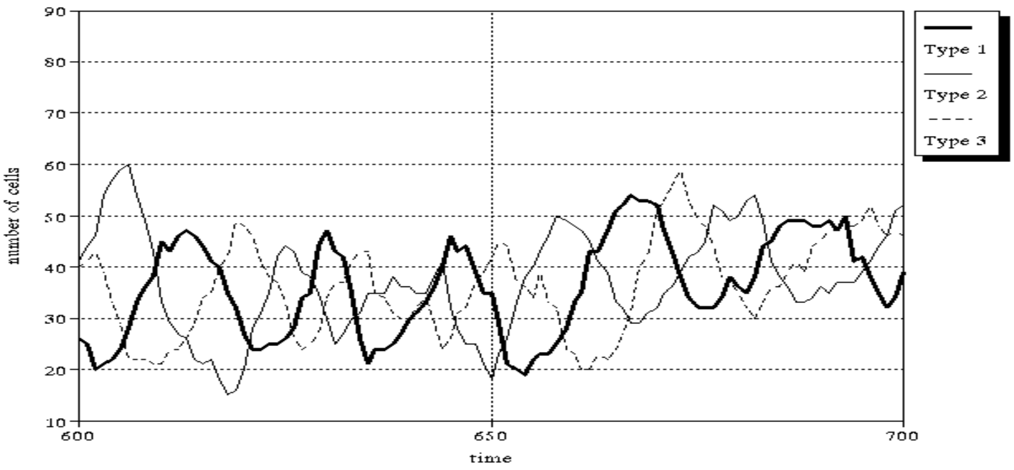
Konglomerát je obecně nekonečná síť konečných automatů [1]. Agregát je pak konglomerát s proměnlivými spoji. Zde je zřetelná paralela s Buněčnou sítí. Buňky sítě by však musely nabývat jen konečného počtu stavů a jejich počet by musel být konstantní.

Ostatní

Z dalších nějak příbuzných přístupů je možné jmenovat třeba genetické algoritmy, buněčné automaty, Valiantův neuroid [5] nebo i poněkud filosofický model myslící polévky. Buněčnou síť však většina těchto modelů připomíná bohužel pouze názvem.

5. Závěrem

V první fázi se podařilo nastavit model tak, aby byl stabilní. Počty buněk v jednotlivých typech se mění periodicky kolem nějaké střední hodnoty. V síti se pohybují buňky tří typů a váhy mezi nimi jsou nastaveny symetricky a činnost modelu připomíná systém lišky-zajíci. [6]



Obrázek 7: Detail činnosti modelu v klidu

Základem vývoje algoritmu učení je tato úvaha podpořená biologickou skutečností.

1. síť je v rovnováze
2. na buňky sítě působí jistý signál
3. síť reaguje a za čas t1 se ustaví nová rovnováha. Během této fáze probíhá učení.
4. nové působení téhož signálu. 5. ustavení rovnováhy v čase t2 « t1

Další naší snahou je popsat stabilitu sítě. Je zřejmé, že stabilita bude nějak záviset na matici vah modelu.

Otázky výpočetní síly přímo souvisejí problémem použití modelu.

References

[1] GOLDSCHLAGER, L.M.: A universal interconnection pattern for parallel computers. *J. of the Assoc. for Comp. Machinery* , Vol.29 **3**, 1073-1086 (1982)

[2] STITES, D.P., TERR, A.I.: *Basic and Clinical Immunology.* Applenton & Lange, A Publishing Division of Prentice Hall, (1991)

[3] *Pokroky v imunologii.* Univerzita Karlova, Sekce postgraduálního studia v biomedicíně (1994)

[4] FRITZKE, B.: Growing cell structures - a self organizing network for unsupervised and supervised learning. *Neural Networks*, Vol.7 **9**, 1441-1460 (1994)

[5] VALIANT, L.G.: *Circuits of the mind.* Oxford University Press, New York, Oxford, (1994)

[6] ŽÁK, P.: Immune Network - Model Inspired by Immune System. *Neural Network World*, Vol.7, **6**, 739-756 (1997)

[7] ŽÁK, P.: Cell Network - Dynamic Neural Network. Learning Problems. *In: EUFIT'98. 6th Europien Congress on Intelligent Techniques and Softcomputing*, Vol.: 1. - Aachen, Mainz 1998, 323-325 (Held: EUFIT'98, Aachen, DE, 98.09.07-98.09.10)

Hybrid Methods of Artificial Intelligence

doktorand:
PAVEL KRUŠINA
Institute of Computer Science AV ČR, P.O. Box 5 182 07
krusina@uivt.cas.cz

školitel:
ROMAN (RO) NERUDA
Institute of Computer Science AV ČR, P.O. Box 5 182 07
roman@uivt.cas.cz

obor studia:
Theoretical computer science

Abstrakt

This paper shows the concept of combining several methods of modern artificial intelligence, namely neural networks, genetic algorithms, and fuzzy logic. We have created a unified software platform — a system called Bang — that allows for easy combination of basic methods encapsulated in blocks. Several experiments and ideas for future work are presented.

1. Introduction

Modern artificial intelligence methods — such as genetic algorithms, neural networks or fuzzy logic — have brought new paradigms to computer science. They are inspired by nature: artificial neural networks model basic concepts of organisms brain functions; genetic algorithms are based on our understanding of evolution and natural selection; fuzzy logic came out has been inspired by natural language.

These methods have several common features. They usualy deal with uncertain and approximate values. They seek for suboptimal solutions of highly non-linear problems possesing multiple local extremes. They are typically rather heuristic procedures employing some random element. They are often able to be trained from exemples. Their knowledge representation is sometimes implicit such as in the case of parameters (weights) of neural networks.

These methods accomplished many successful applications in real world problems such as classification, control, pattern recognition, optimalization etc. However, they also have their weak sides such as the lack of theoretical background and deep understanding. We also miss a unifying concept encompassing various methods and allowing their easy combinations.

2. The unified framework

Recently, a term *soft computing* has been proposed by L. Zadeh with the meaning of creative fussion of neural networks, genetic algorithms and fuzzy logic.

In our terminology, we introduce the *building block* as a highly encapsulated object, which realizes a particular method (either AI or not). Typically a block can have several inputs and/or outputs, and it possesses some inner parameters determining its functionality. The emphasis is laid on the openness, modularity and interchangability of the blocks. The principle of hiding as much of the blocks' individuality as possible allows and highly encourages interchange of the blocks. As long as the interface between blocks remains stable, we can replace one building block with another without any impact to the rest of the blocks. If we have a building block as an abstraction of a method, then the only thing a block interface have to consist of is some mechanism of data exchange. Gate is a data flow entry point of the building block. In fact, the only thing the block interface consists of is a constant set of gates. In order that input gate is able to receive data and an output gate is able to send them, they have to be connected. The connection of an input gate and an output one is called channel. Each channel has a selected data type and only data of this type can be passed through.

3. The Bang

Our first attempt on this field was the Bang system. The basic requirements on the Bang program are to be portabable, extensible and user friendly. Mainly the "being user friendly" part, which basically means to have a good graphical user interface (UI), and to be portable, is difficult to fulfill. This leads to architecture of highly independent modules, which in turn reduces the code rewriting to a low level. An interesting implication of these design principles is the fact that there is not just one Bang program. One can rather combine individual modules to customize Bang functionality.

3.1. Modules

The Bang program is a composition of an executive core (which does the real work), an abstract user interface (which can be window or console oriented and does all the communication with the user in an abstract and portable manner) and a user interface implementation module (which is the highly platform dependent part of the user interface).

Each Bang program consists of the building blocks, the engine, an abstract user interface and a user interface library wrapper. It may also contain some graphics library wrappers.

- Window oriented style

  The basic and most user-friendly configuration of Bang takes the advantage of modern graphics user interface systems like X Window System or Microsoft Windows.

- Console oriented style

  If you are not so lucky to have a windowing system on your computer, you are forced to use the console oriented interface.

- Frozen SEN

  This configuration of Bang is designed for a batch style usage. This form of Bang program performs no interactions with user, all information it needs is hardwired to the executable file itself. To make this work you must first include the sen and ins files (sen file describes the building blocks schema and ins file stores initialization data for these blocks).

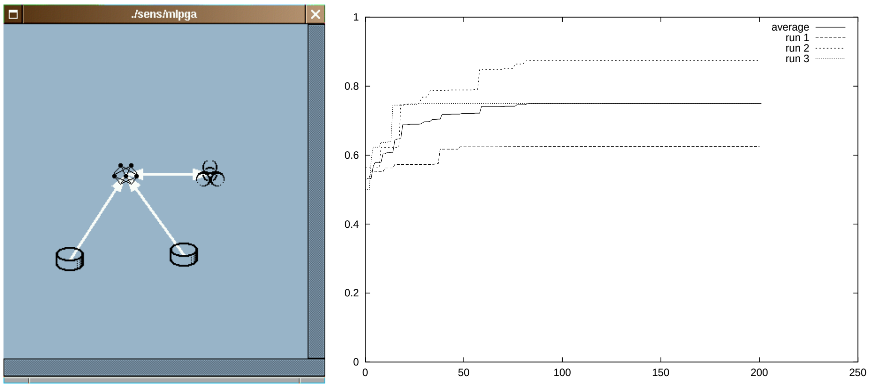
4. Experiments

4.1. Triparity

The problem we have choosen as a reference base for our tests is usualy called triparity. The triparity function has tree binary inputs and one binary output — the input's parity.

4.2. MLP + GA

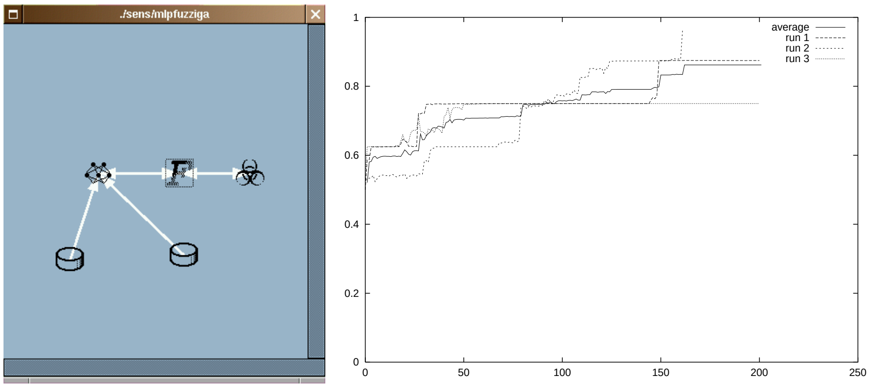
In this experiment, we have turn off the back-propagation and used genetic algorithm instead (Figure 8). The mutation and crossover rates were set to 0.4 and 0.5, the target fitness level was 0.95. The population consisted of 40 genoms — vectors of 32 — the number of weights in 3-5-2 MLP — numbers from -5 to $+5$. Each generation change needs 40 MLP evaluations.



Obrázek 8: MLP + GA

4.3. MLP + Fuzzifier + GA

In this experiment, we put a fuzzifier block in between MLP and GA (Figure 9). Both MLP and GA were configured in the same way as in the previous case, the fuzzifier block was set to fuzzy input genoms — from GA — into 5 samples of their fuzzifications each. This configuration requires 200 (5 times 40) MLP evaluations per generation.



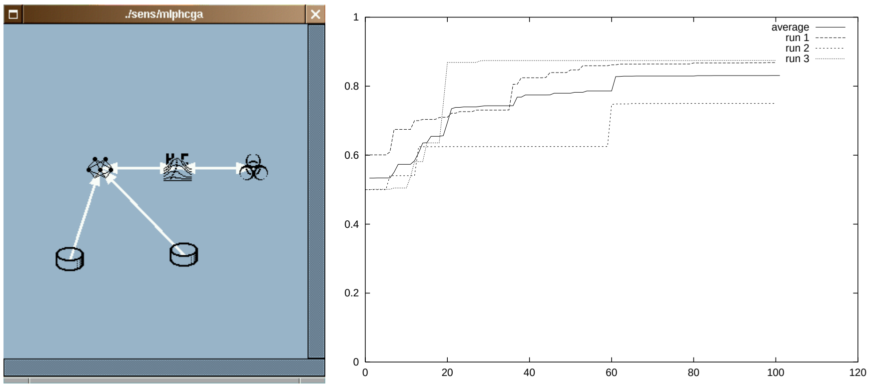
Obrázek 9: MLP + Fuzzifier + GA

4.4. MLP + HC + GA

This time, we exchanged the fuzzifier block with the hill climbing one (Figure 10). The MLP and GA keeped their configurations, the hill climbing was set to do three steps with each genom from GA. Unfortunately each HC step needs 64 evaluations of MLP — a small step in both directions of all 32 dimensions. This leads to 7680 MLP evaluations per each generation step (64 times 3 times 40).

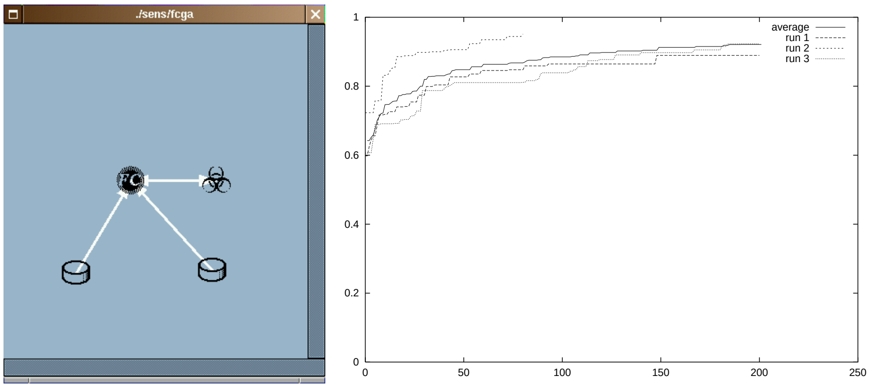
4.5. FLC + GA

The next experiment was the fuzzy logic controller tought by genetic algorithm (Figure 11). The mutation and crossover rates were set to 0.5 and 0.6, the target fitness level was 0.95. The population consisted of 40 genoms — vectors of 96 numbers from 0 to +1. The number 96 came from 3 samples per fuzzy set membership function multiplied by 4 for three inputs and one output per rule and finally multiplied by 8 for 8 rules in a ruleset. Each generation step needs 40 FLC



Obrázek 10: MLP + HC + GA

evaluations. Note that because of another size of genom and the whole difference between FLC and MLP, there is no evident relationship between this numbers and time measured in MLP evaluations.



Obrázek 11: FLC + GA

4.6. MLP + FLC

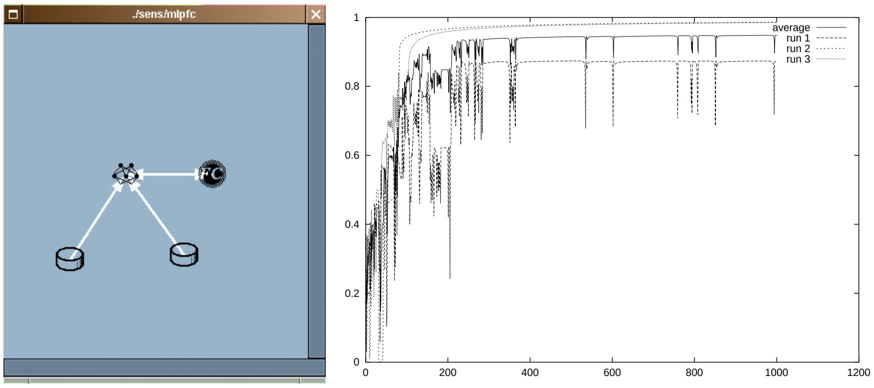
In this experiment we have turned the back-propagation on and let the fuzzy logic controller to tune its ε — the learning speed — and α — the momentum — parameters (Figure 12). The time complexity is similar to the undriven back-propagation learning case — 2 MLP evaluations for each BP step.

4.7. MLP + GA + FLC

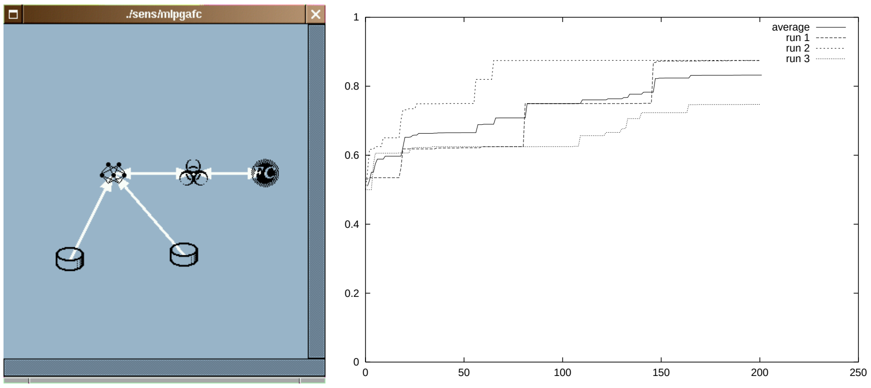
In this experiment, we have turned off the back-propagation and used genetic algorithm instead (Figure 13). The target fitness level was set 0.95 and the mutation and crossover rates were tuned by fuzzy logic controller. The population consisted of 40 genoms — vectors of 32 numbers from -5 to $+5$. Each generation change needs 40 MLP evaluations – same as in the MLP + GA case because using FLC adds no additional time complexity.

4.8. FLC + HC

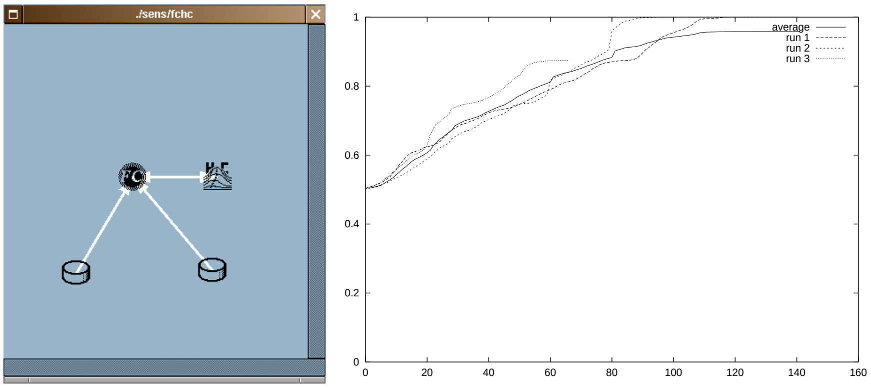
This is a test of teaching fuzzy logic controller via the hill climbing technique (Figure 14). Because of the fact that the FLC is determined in this configurations by a vector of 96 numbers, it takes 192 FLC evaluations to do a single HC step.



Obrázek 12: MLP + FLC



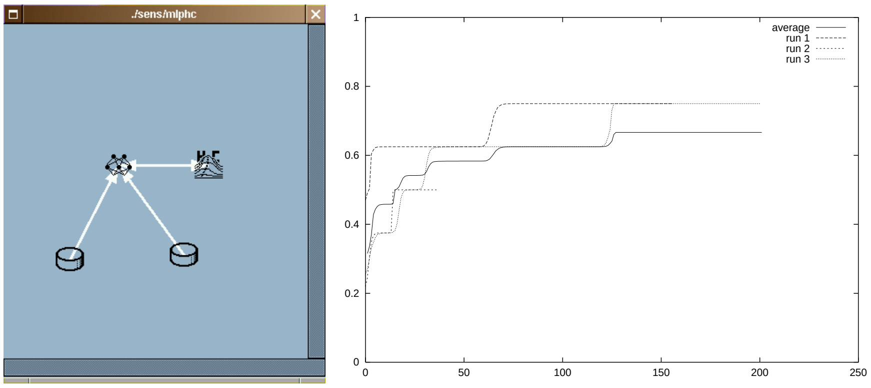
Obrázek 13: MLP + GA + FLC



Obrázek 14: FLC + HC

4.9. MLP + HC

In the last experiment, we tought the MLP via the hill climbing (Figure 15). In the case of MLP it takes 64 evaluations to do a HC step.



Obrázek 15: MLP + HC

4.10. Conclusion

Let us draw several conclusions based on the above described experiments. Basically, we deal with three different problem areas. The first is learning fuzzy logic controllers in which case genetic algorithms (4.5) outperformed hill climbing (4.8). The second is using fuzzy logic controllers for tuning learning parameters of other methods. It shows that FLC can really improve performance of both GAs (4.7) and BP (4.6). Finally, comparing all methods together, we can conclude that the fuzzy controlled back-propagation is the overall winner (4.6).

5. Further work

The following ideas will be considered in an attempt to create more powerful and innovative version of Bang.

- building block being a software agent or at least having some agent-specific features.

  The building block is an encapsulation of an AI method. It is the cooperation of building block instances which creates a true hybrid systems. It makes a good sense to give some amount of artificial intelligence to these objects. In Bang we have modelled artificial intelligence tasks using classical non-AI objects. Now we will try to use artificial intelligence to assist us with the task of testing hybrid artificial intelligence methods.

- building block being a regular CORBA component

  In Bang, the issues of portability and effectivness avoided using some implementation of CORBA as the virtual environment, where our building block object lived. Maybe, now is the time to sacrifice some of the effectivness for gain in other fields.

- dynamic schema modifications maybe initiated by building blocks themselves

  Ideas from [1] seems to add another dimension in our thoughts about the hybrid artificial intelligence methods and the way the methods can cooperate. We want to try making the whole system able of run-time building block instances creation, connection and reconfiguration. Today it seems to be possible to create a building block responsible for such actions in some section of building blocks schema. But there is much work to be done, before we will be able to really program such a system.

- automatic schema testing and comparing

  The idea is to give the test data and schema set to the system and let it do series of test or benchmarks on its own. Maybe some kind of scripting mechanism will be useful for the results evaluation — automatic creating of gnuplot graphs or tex tables comes me to mind.

References

[1] Amanda J.C. Sharkey (Ed.), "Combining Artificial Neural Nets", *Springer*, 1999.

[2] Pavel Krušina, "Hybrid Methods of Artificial Intelligence in Data Analysis", 1999.

[3] Melanie Mitchell, "An Introduction to Genetic Algorithms", *The MIT Press*, 1996.

[4] Simon Haykin "Neural Networks", *IEEE Press*, 1994.

[5] Gerhard Weiss (Ed.), "Multiagent Systems", *The MIT Press*, 1999.