Doktorandský den ’05

Ústav informatiky
Akademie věd České republiky

Hosty – Týn nad Vltavou

5. – 7. říjen 2005

vydavatelství Matematicko-fyzikální fakulty
University Karlovy v Praze

ISBN *– not yet –*

Obsah

Doktorandský den Ústavu informatiky Akademie věd České republiky se koná již podesáté, nepřetržitě od roku 1996. Tento seminář poskytuje doktorandům, podílejícím se na odborných aktivitách Ústavu informatiky, možnost prezentovat výsledky jejich odborného studia. Současně poskytuje prostor pro oponentní připomínky k přednášené tematice a použité metodologii práce ze strany přítomné odborné komunity.

Z jiného úhlu pohledu, toto setkání doktorandů podává průřezovou informaci o odborném rozsahu pedagogických aktivit, které jsou realizovány na pracovištích či za spoluúčasti Ústavu informatiky AV ČR.

Jednotlivé příspěvky sborníku jsou uspořádány podle jmen autorů. Uspořádání podle tematického zaměření nepovažujeme za účelné, vzhledem k rozmanitosti jednotlivých témat.

Vedení Ústavu informatiky a Vědecká rada ÚI jakožto organizátor doktorandského dne věří, že toto setkání mladých doktorandů, jejich školitelů a ostatní odborné veřejnosti povede ke zkvalitnění celého procesu doktorandského studia zajišťovaného v součinnosti s Ústavem informatiky a v neposlední řadě k navázání a vyhledání nových odborných kontaktů.

1. září 2005

.... nejprve se obraťme k ústřednímu dílu staré čínské matematické literatury, k “Matematice v devíti knihách” (“Ťiou čang suan šu”)[1]. V tomto traktátu byly shrnuty výsledky mnohaleté práce matematiků, kteří žili v 1. tisíciletí před n. l. Tento traktát je nejstarší z dochovaných čínských spisů, věnovaných výhradně matematice. Jeho jazykem je starověká čínština, značně odlišná od současné spisovné čínštiny.

Přesná doba vzniku, prameny a autoři “Matematiky v devíti knihách” nejsou známi. Liou Chuej, který ve 3. století “Matematiku v devíti knihách” komentoval, uvádí, že ji podle starších spisů vytvořil významný úředník finanční služby Čang Chang, který vykonával řadu let funkci prvého ministra. Čang Chang podle staročínských kronik zemřel roku **152** před n.l. Tentýž Liou Chuej píše, že přibližně o sto let později knihu přepracoval jiný vysoký úředník a ministr Keng Šou-čchang; jeho působnost spadá do období vlády imperátora Süan-ti (**73 – 49** před n.l.). ....

.... metoda fang čcheng, probíraná v 8. knize “Matematiky v devíti knihách”, je bezesporu největším objevem čínských matematiků, zabývajících se řešením soustav lineárních úloh. Tato metoda udává algoritmus řešení systému n lineárních rovnic o n neznámých. Použijeme-li moderní symboliku, pak metoda fang čcheng je přímou metodou pro řešení systému rovnic

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & = & b_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1}x_1 & + & a_{n2}x_2 & + & \cdots & + & a_{nn}x_n & = & b_n \end{array} .$$

Tento systém rovnic je pomocí tabulky fang čcheng vyjádřen na počítací desce, přičemž koeficienty jednotlivých rovnic se přenášejí do tabulky shora dolů a rovnice jsou “zapisovány” zprava doleva:

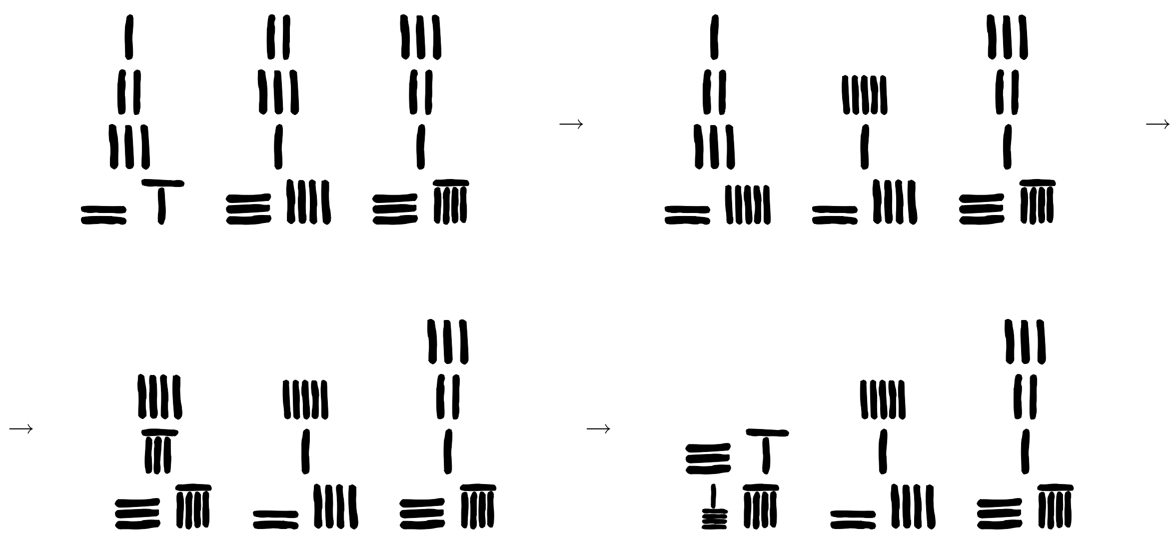
$$\begin{array}{cccc} a_{n1} & \cdots & a_{21} & a_{11} \\ a_{n2} & \cdots & a_{22} & a_{12} \\ \cdots & \cdots & \cdots & \cdots \\ a_{nn} & \cdots & a_{2n} & a_{1n} \\ b_n & \cdots & b_2 & b_1 \end{array} .$$

[1]Převzato z A.P.Juškevič, Dějiny matematiky ve středověku, Academia, Praha, 1977.

Tato tabulka se začne upravovat odečtením prvého sloupce zprava od prvků druhého sloupce, který byl předtím vynásoben prvkem a_{11}, tak dlouho dokud na místě prvku a_{21} nezbude nic[2]. Tentýž postup aplikujeme i na další sloupce, takže první řádek bude kromě prvku a_{11} obsahovat jen prázdná místa. Obdobně vyprázdníme druhý řádek tabulky, dokud nezískáme tabulku, jejíž levý horní roh je prazdný. Ukažme si názorně toto obecné schéma na první úloze ze 7. knihy, na základě které bylo pravidlo fang čcheng formulováno:

3 snopy z dobré úrody, 2 snopy z průměrné úrody a jeden snop ze špatné úrody dávají 39 tou zrní; 2 snopy z dobré úrody, 3 z průměrné a 1 ze špatné dávají 34 tou; 1 snop z dobré, 2 snopy z průměrné a 3 ze špatné dávají 26 tou. Ptáme se, kolik zrní dává každý snop z dobré, průměrné a špatné úrody?

Odpovídající posloupnost tabulek fang čcheng po jednotlivých úpravách je následující [3]



.... úpravy tabulky fang čcheng jsou v podstatě operace s maticemi a determinanty, kterým dnes říkáme Gaussova eliminace. Metoda fang čcheng byla v Orientu dále rozvinuta a nakonec vyústila ve svéráznou teorii determinantů, především v rukopise japonského matematika Seki Šinsuke (Kowa) z roku 1683. V Evropě se s metodou přímého řešení lineárních rovnic sestkáváme poprvé u Leonarda Pisánského a G. Cardana (**1545**). Zcela jasně však formuloval myšlenku zavést determinanty při eliminaci neznámých až Leibnitz v dopise l'Hospitalovi r. **1693**. Později ji pak rozpracoval a při řešení lineárních systémů použil Gabriel Cramer (**1750**). ....

---

[2] Koeficienty v úlohách "Matematiky v devíti knihách" jsou celá kladná čísla. V důsledku postupného odečítání sloupců se v obecné lineární úloze musí využívat záporná čísla. V metodě fang čcheng se prvně v historii zavadí pojem záporných čísel.

[3] V době sepisování "Matematiky v devíti knihách" používali v Číně desítkovou poziční soustavu, kterou vyjadřovali 18 číslicemi 1 – 9 (𝍠 ... 𝍤 , 𝍥 ... 𝍨) a 10 – 90 (𝍩 ... 𝍭 , 𝍮 ... 𝍱). Znak pro nulu chyběl, nahrazoval se prázdným místem. Čísla se zapisovala tak, že jednotky na 3-tím místě zprava označovaly stovky, desítky na 4-tém místě zprava označovaly tisíce atd. Záporná čísla se označovala položením tyčinky šikmo přes poslední číslici.

Fuzzy MacNeille and Dedekind Completions of Crisp Dense Linear Orderings

Post-Graduate Student:
MGR. LIBOR BĚHOUNEK

Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2

182 07 Prague 8

behounek@cs.cas.cz

Supervisor:
DOC. PHDR. PETR JIRKŮ, CSC.

Faculty of Philosophy
Charles University in Prague
Celetná 20
116 42 Prague 1

Czech Republic

petr.jirku@ff.cuni.cz

Field of Study:
Logic
Classification: 6101 V

This work was supported by grant No. GD-401/03/H047 of the Grant Agency of the Czech Republic *Logical foundations of semantics and knowledge representation.* The co-advisor for my research in the area of fuzzy logic is Prof. RNDr. Petr Hájek, DrSc.

Abstract

In the framework of Henkin-style higher-order fuzzy logic we define two kinds of the fuzzy lattice completion. The fuzzy MacNeille completion is the lattice completion by (possibly fuzzy) stable sets; the fuzzy Dedekind completion is the lattice completion by (possibly fuzzy) Dedekind cuts. We investigate the properties and interrelations of both notions and compare them to the results from the literature. Our attention is restricted to crisp dense linear orderings, which are important for the theory of fuzzy real numbers.

1. The framework

In [1] and [2], Henkin-style higher-order fuzzy logic is proposed as a foundational theory for fuzzy mathematics. In this framework, and following the methodology of [1], we define and investigate two notions of fuzzy lattice completion of crisp dense linear orderings: the MacNeille completion and the Dedekind completion. Both methods generalize the (classically equivalent) constructions of complete lattices by admitting fuzzy sets into the completion process. The theory of the fuzzy lattice completion is important for the construction of fuzzy real numbers within the formal framework for fuzzy mathematics.

For the ease of reference we repeat here the definitions of the Henkin-style higher-order fuzzy logic presented in [2], generalized here to any fuzzy logic containing the first-order logic BL_Δ. See [3] for BL_Δ, and [4] for first-order fuzzy logics with function symbols.

Definition 1 (Henkin-style higher-order fuzzy logic) *Let $\mathcal{F}$ be a fuzzy logic which extends BL_Δ. The* Henkin-style higher-order fuzzy logic over $\mathcal{F}$ *(denoted here by $\mathcal{F}_\omega$) is a theory over multi-sorted first-order $\mathcal{F}$ with the following language and axioms:*

For all finite n, there is the sort of fuzzy sets of the n-th order, which subsumes the sorts of k-tuples of fuzzy sets of the n-th order (for all finite k). Fuzzy sets of the 0-th order can be regarded as atomic objects; 1-tuples of fuzzy sets can be identified with the fuzzy sets themselves; and we can assume that for all m, n

such that $m > n$, the sort of fuzzy sets of order n is subsumed in the sort of fuzzy sets of order m. Variables of sort n are denoted by $x^{(n)}, y^{(n)}, \ldots$; there are no universal variables. We omit the upper index if the order is arbitrary or known from the context.

The language of the theory contains:

- *The function symbols for tuple formation and component extraction. We shall use the usual notation $\langle x_1, \ldots, x_k \rangle$ for k-tuples.*
- *Comprehension terms $\{x \mid \varphi\}$, for any formula φ. If x is of order n, then $\{x \mid \varphi\}$ is of order $n + 1$.*
- *The identity predicate $=$ (for all sorts).*
- *The membership predicate $\in$ between orders n and $n + 1$.*

The axioms of $\mathcal{F}_\omega$ are the following, for all orders:

1. *The axioms of identity: the reflexivity axiom $x = x$ and the intersubstitutivity schema $x = y \rightarrow (\varphi(x) \rightarrow \varphi(y))$.*
2. *The axioms of tuples: tuple formation and component extraction are inverse operations; tuples equal iff all their components equal.*
3. *The comprehension schema $y \in \{x \mid \varphi(x)\} \leftrightarrow \varphi(y)$, for any formula φ.*
4. *The extensionality axiom $(\forall x)\Delta(x \in X \leftrightarrow x \in Y) \rightarrow X = Y$.*

The intended models of the theory are Zadeh's fuzzy sets of all orders on a fixed domain. For further technical details see [2].

We shall freely use all abbreviations common in classical mathematics, and assume the usual precedence of logical connectives. Unless stated otherwise, definitions and theorems apply to all orders of fuzzy sets (in this paper, however, we only employ the first three orders; thus in fact we work in the third-order BL_Δ).

Convention 2 *We shall denote atomic objects by lowercase variables, fuzzy sets of atomic objects by uppercase variables, and second-order fuzzy sets by calligraphic variables.*

Definition 3 *We shall need the following defined concepts:*

Id	$=_{\mathrm{df}}$	$\{\langle x, y\rangle \mid x = y\}$	*identity relation*
$\mathrm{Ker}(X)$	$=_{\mathrm{df}}$	$\{x \mid \Delta(x \in X)\}$	*kernel*
$X \cap Y$	$=_{\mathrm{df}}$	$\{x \mid x \in X \mathbin{\&} x \in Y\}$	*pair intersection*
$\mathcal{P}(X)$	$=_{\mathrm{df}}$	$\{Y \mid Y \subseteq X\}$	*power set*
$\bigcap \mathcal{A}$	$=_{\mathrm{df}}$	$\{x \mid (\forall A \in \mathcal{A})(x \in A)\}$	*set intersection*
$\bigcup \mathcal{A}$	$=_{\mathrm{df}}$	$\{x \mid (\exists A \in \mathcal{A})(x \in A)\}$	*set union*
$X \subseteq Y$	$\equiv_{\mathrm{df}}$	$(\forall x)(x \in X \rightarrow x \in Y)$	*inclusion*
$X \approx Y$	$\equiv_{\mathrm{df}}$	$(\forall x)(x \in X \leftrightarrow x \in Y)$	*bi-inclusion*
$\mathrm{Refl}(R)$	$\equiv_{\mathrm{df}}$	$(\forall x)Rxx$	*reflexivity*
$\mathrm{Trans}(R)$	$\equiv_{\mathrm{df}}$	$(\forall xyz)(Rxy \mathbin{\&} Ryz \rightarrow Rxz)$	*transitivity*
$\mathrm{ASym}_E(R)$	$\equiv_{\mathrm{df}}$	$(\forall xy)(Rxy \mathbin{\&} Ryx \rightarrow Exy)$	*E-antisymmetry*
$\mathrm{Fnc}_E(R)$	$\equiv_{\mathrm{df}}$	$(\forall xyz)(Rxy \mathbin{\&} Rxz \rightarrow Eyz)$	*E-functionality*

By convention, the index E can be dropped if $\Delta(E = \mathrm{Id})$; if $\Delta\,\mathrm{Fnc}(F)$, we can write $y = F(x)$ instead of ΔFxy.

2. Cones, suprema, and infima

We fix an arbitrary binary fuzzy relation $\leq$. Although the following notions are most meaningful for fuzzy quasi-orderings (i.e., reflexive and transitive relations), we need not impose any restrictions on $\leq$.

Definition 4 *The set A is* upper *(in $\leq$) iff $(\forall x, y)[x \leq y \rightarrow (x \in A \rightarrow y \in A)]$. Dually, the set A is* lower *iff $(\forall x, y)[x \leq y \rightarrow (y \in A \rightarrow x \in A)]$.*

Definition 5 *The* upper cone *and the* lower cone *of a set A (w.r.t. $\leq$) are respectively defined as follows:*

$$\begin{aligned} A^{\uparrow} &=_{\mathrm{df}} \{x \mid (\forall a \in A)(a \leq x)\} \\ A^{\downarrow} &=_{\mathrm{df}} \{x \mid (\forall a \in A)(x \leq a)\} \end{aligned}$$

Lemma 6 *The following properties of cones (known from classical mathematics for crisp sets) can be proved in $\mathcal{F}_\omega$ (we omit the dual versions of the theorems):*

1. $A \subseteq B \rightarrow B^{\uparrow} \subseteq A^{\uparrow}$ *(antitony w.r.t. inclusion)*
2. $A \subseteq A^{\uparrow\downarrow}$ *(closure property)*
3. $A^{\uparrow\downarrow\uparrow} = A^{\uparrow}$ *(stability)*
4. $\mathrm{Trans}(\leq) \rightarrow A^{\uparrow}$ *is upper*

The usual definition of suprema and infima as least upper bounds and greatest lower bounds can then be formulated as follows:

Definition 7 *The sets of all suprema and infima of a set A w.r.t. $\leq$ are defined as follows:*

$$\begin{aligned} \mathrm{Sup}\, A &=_{\mathrm{df}} A^{\uparrow} \cap A^{\uparrow\downarrow} \\ \mathrm{Inf}\, A &=_{\mathrm{df}} A^{\downarrow} \cap A^{\downarrow\uparrow} \end{aligned}$$

Lemma 8 *The following properties of suprema (known from classical mathematics for crisp sets) can be proved in $\mathcal{F}_\omega$ (we omit the dual theorems for infima):*

1. $(A \subseteq B \mathbin{\&} x \in \mathrm{Sup}\, A \mathbin{\&} y \in \mathrm{Sup}\, B) \rightarrow x \leq y$ *(monotony w.r.t. inclusion)*
2. $(x \in \mathrm{Sup}\, A \mathbin{\&} y \in \mathrm{Sup}\, A) \rightarrow (x \leq y \mathbin{\&} y \leq x)$ *(uniqueness)*
3. $\mathrm{Sup}\, A = \mathrm{Inf}\, A^{\uparrow}$ *(interdefinability)*

Notice that $\mathrm{Sup}\, A$ and $\mathrm{Inf}\, A$ are fuzzy sets, since the property of being a bound in a fuzzy ordering is generally fuzzy. Nevertheless, if $\leq$ is antisymmetric w.r.t. relation E, then by 2. of Lemma 8, the suprema and infima are E-unique. If furthermore $\mathrm{Ker}(E)$ is identity, the unique element of $\mathrm{Ker}(\mathrm{Sup}\, A)$ can be called *the* supremum of A and denoted by $\sup A$.

Example 1 *$\bigcup \mathcal{A}$ is a supremum of $\mathcal{A}$ w.r.t. $\subseteq$. By 2. of Lemma 8, the suprema w.r.t. $\subseteq$ are unique w.r.t. bi-inclusion $\approx$. Due to the extensionality axiom, the element of the kernel of $\mathrm{Sup}_{\subseteq} \mathcal{A}$ is unique w.r.t. identity; thus $\sup_{\subseteq} \mathcal{A} = \bigcup \mathcal{A}$. (Dtto for $\bigcap$ and infima.)*

Definition 9 *A is* lattice complete $\equiv_{\mathrm{df}} (\forall X \subseteq A)(\exists x \in A)(x \in \operatorname{Sup} X)$.

Thus A is lattice complete in the degree 1 iff all fuzzy subsets of A have 1-true suprema in A. The existence of 1-true infima then already follows by 3. of Lemma 8

Example 2 *Due to Example 1, the power set $\mathcal{P}(A) =_{\mathrm{df}} \{X \mid X \subseteq A\}$ is lattice complete w.r.t. $\subseteq$.*

3. Fuzzy lattice completions of dense linear crisp orders

Further on, we restrict our attention to *linear crisp* domains, since we aim (cf. [5]) at constructing a formal theory of fuzzy numbers in the usual sense, i.e. based on some system of crisp numbers (integer, rational, or real). The theory of fuzzy lattice completions of linear dense crisp domains is an important part of this enterprise, as we want the resulting system of fuzzy real numbers or intervals to be lattice complete. It turns out that discrete domains behave quite differently under fuzzy lattice completions, so we shall only consider *dense* domains here.

We distinguish two methods of fuzzy lattice completion of crisp linear dense domains, which generally differ in fuzzy logic (unlike classical logic): *Dedekind* completion by fuzzy Dedekind cuts (lower right-closed sets), and *MacNeille* completion by fuzzy stable sets. Both methods directly generalize the classical Dedekind-MacNeille completion by admitting fuzzy sets to be involved in the process. Both methods yield complete lattices and preserve existing suprema and infima. However, the resulting lattices cannot generally be characterized as the least complete lattices extending the original order. (The latter is, of course, the crisp Dedekind-MacNeille completion, as we start with a crisp order; they are just the least completions containing all fuzzy cuts or all fuzzy stable sets.)

In crisp linear dense orderings, cones have a special property that will be used later:

Lemma 10 *If $\leq$ is a crisp linear dense ordering, then $y \in A^{\uparrow} \equiv (\forall a > y)\neg(a \in A)$.*

3.1. The fuzzy MacNeille completion

We define the fuzzy MacNeille completion for lower stable sets. The construction can of course be dualized for upper sets.

Definition 11 *We call A a* (lower) stable set *iff $A^{\uparrow\downarrow} = A$.*

Definition 12 *The* MacNeille fuzzy lattice completion *$\mathcal{M}(X)$ of X is the set of all stable subsets of X, ordered by inclusion.*

Observation 13 *All lower stable sets are lower in the sense of Definition 4. The property of being a stable set is crisp, since $=$ is a crisp predicate in $\mathcal{F}_\omega$. Therefore, $\mathcal{M}(X)$ is a crisp set of (possibly fuzzy) sets.*

Theorem 14 *$\mathcal{M}(X)$ is lattice complete. X is embedded in $\mathcal{M}(X)$ by assigning $\{x\}^{\downarrow}$ to $x \in X$; the embedding preserves all suprema and infima that already existed in X. The suprema and infima in $\mathcal{M}(X)$ are unique w.r.t. bi-inclusion. Due to the extensionality axiom, there is a unique* $\sup \mathcal{A} \in \operatorname{Ker}(\operatorname{Sup} \mathcal{A})$ *for any $\mathcal{A} \in \mathcal{M}(X)$ (dtto for infima). Furthermore,* $\inf \mathcal{A} = \bigcap \mathcal{A}$ *and* $\sup \mathcal{A} = (\bigcup \mathcal{A})^{\uparrow\downarrow}$*, as in classical mathematics.*

3.2. The fuzzy Dedekind completion

We define the fuzzy Dedekind completion for lower Dedekind cuts. The construction can of course be dualized for upper cuts.

Definition 15 *We call A a* (lower) Dedekind cut *iff it satisfies the following two axioms:*

$$\Delta(\forall x, y)[x \leq y \rightarrow (y \in A \rightarrow x \in A)]$$
$$\Delta(\forall x)[(\forall y < x)(y \in A) \rightarrow x \in A]$$

Thus fuzzy Dedekind cuts are lower, right-closed subsets of X, i.e., their membership functions are non-increasing and left-continuous. The conditions reflect the intuitive motivation that the membership $x \in A$ expresses (the truth value of) the fact that x *minorizes* the "fuzzy element" A: the first axiom then corresponds to the transitivity of the minorization relation, and the second axiom to the minorization in the sense of $\leq$ rather than $<$. Since any flaw in monotony or left-continuity makes A strictly violate the motivation, the axioms are required to be 1-true (by the Δ's in the definition); the property of being a Dedekind cut is therefore crisp (although the Dedekind cuts themselves can be fuzzy sets).

Definition 16 *The* fuzzy Dedekind completion $\mathcal{D}(X)$ *of X is the set of all Dedekind cuts on X, ordered by inclusion.*

The properties of the MacNeille completion listed in Theorem 14 hold for $\mathcal{D}(X)$, too. Considering that in the embedding of X into $\mathcal{D}(X)$, the cone $\{x\}^{\downarrow}$ is the counterpart of $x \in X$ in $\mathcal{D}(X)$ and $\subseteq$ on $\mathcal{D}(X)$ corresponds to $\leq$ on X, the following theorem proves the soundness of the axioms of Dedekind cuts w.r.t. the intuitive motivation above:

Theorem 17 *For all Dedekind cuts $A \in \mathcal{D}(X)$ and for all $x \in X$ it holds that $x \in A \leftrightarrow \{x\}^{\downarrow} \subseteq A$.*

4. A comparison of the two notions

In classical mathematics, $\mathcal{M}(X) = \mathcal{D}(X)$. In $\mathcal{F}_{\omega}$, only one inclusion can be proved generally:

Theorem 18 *Any stable set is a Dedekind cut. Therefore, $\mathcal{M}(X) \subseteq \mathcal{D}(X)$.*

The converse inclusion does not generally hold. If the negation is strict (i.e., $\neg\varphi \vee \neg\neg\varphi$ holds, as e.g. in Gödel or product logic), all cones (and therefore, all stable sets) are crisp by Lemma 10. Thus in the logic SBL_{Δ} (i.e., BL_{Δ} with strict negation) or stronger (e.g., G_{Δ}, Π_{Δ}, or classical logic), the fuzzy MacNeille completion coincides with the crisp MacNeille completion. The fuzzy Dedekind completion of non-empty sets, on the other hand, always contains non-crisp Dedekind cuts (in non-crisp models of $\mathcal{F}_{\omega}$).

In Łukasiewicz logic Ł_{Δ} (or stronger), the properties of fuzzy completions are much closer to properties of the classical Dedekind–MacNeille completion than in other fuzzy logics (this is due to the involutiveness of Łukasiewicz negation):

Theorem 19 *In Ł_{Δ}, the notions of Dedekind cuts and stable sets coincide, i.e., $\mathcal{M}(X) = \mathcal{D}(X)$.*

Theorem 20 *$\subseteq$ is a weak linear order on the Dedekind–MacNeille completion, i.e., $(\forall A, B \in \mathcal{D}(X))(A \subseteq B \mathbin{\underline{\vee}} B \subseteq A)$, where $\underline{\vee}$ is the strong (co-norm) disjunction.*

The latter property cannot be proved generally (in logics where co-norm disjunction is present): in particular, it fails for the Gödel co-norm $\vee$ (as the max-disjunction linearity is equivalent to excluded middle).

5. A comparison with results from the literature

Höhle's paper [7] and a chapter in Bělohlávek's book [8] study the minimal completion of *fuzzy* orderings (by the construction that we call the MacNeille completion in the present paper). In our present setting we are, on the other hand, concerned with the lattice completions of *crisp* orders by fuzzy sets. The MacNeille completion of crisp orders is mentioned towards the end of [7]; some of the results of Section 4 on $\mathcal{M}(X)$ in $Ł_\triangle$, $\Pi_\triangle$, and $G_\triangle$ are obtained there.

Both [7] and [8] use slightly different definitions of fuzzy orderings and suprema. In particular, they use the min-conjunction $\wedge$ in the definition of antisymmetry ([8] also in the definitions of the supremum and infimum) instead of strong conjunction $\&$. Such definitions are narrower, and thus the results less general. Reasons can be given why strong conjunction rather than min-conjunction should (from the point of view of formal fuzzy logic) be used in the definitions: the results of [8] are then well-motivated only in Gödel logic. Nevertheless, their results on the suprema and infima in both works are virtually the same as those in Section 2, since most of the properties of the suprema do not depend on the properties of $\leq$. The setting of [7] is further complicated by the apparatus for the accommodation of the fuzzy domains of $\leq$. However, the analogues of Theorem 14 for fuzzy ordering are proved even in the different settings of [7] and [8].

Incidentally, both notions of fuzzy lattice completion defined in the present paper satisfy Dubois and Prade's requirement of [9] that the cuts of fuzzy notions be the corresponding crisp notions. This is a rather general feature of classical definitions transplanted to formal fuzzy logic (which is a general methodology of [2], foreshadowed already in [7]).

References

[1] L. Běhounek and P. Cintula, "From fuzzy logic to fuzzy mathematics: A methodological manifesto." Submitted to Fuzzy Sets and Systems, 2005.

[2] L. Běhounek and P. Cintula, "Fuzzy class theory," *Fuzzy Sets and Systems*, vol. 154, no. 1, pp. 34–55, 2005.

[3] P. Hájek, *Metamathematics of Fuzzy Logic*, vol. 4 of *Trends in Logic*. Dordercht: Kluwer, 1998.

[4] P. Hájek, "Function symbols in fuzzy logic," in *Proceedings of the East-West Fuzzy Colloquium*, (Zittau/Görlitz), pp. 2–8, IPM, 2000.

[5] L. Běhounek, "Towards a formal theory of Dedekind fuzzy reals," in *Proceedings of EUSFLAT*, (Barcelona), 2005.

[6] G. Takeuti and S. Titani, "Intuitionistic fuzzy logic and intuitionistic fuzzy set theory," *Journal of Symbolic Logic*, vol. 49, no. 3, pp. 851–866, 1984.

[7] U. Höhle, "Fuzzy real numbers as Dedekind cuts with respect to a multiple-valued logic," *Fuzzy Sets and Systems*, vol. 24, no. 3, pp. 263–278, 1987.

[8] R. Bělohlávek, *Fuzzy Relational Systems: Foundations and Principles*, vol. 20 of *IFSR Int. Series on Systems Science and Engineering*. New York: Kluwer Academic/Plenum Press, 2002.

[9] D. Dubois and H. Prade, "Fuzzy elements in a fuzzy set" , in Y. Liu, G. Chen, M. Ying (eds.), *Fuzzy Logic, Soft Computing and Computational Intelligence: Eleventh International Fuzzy Systems Acssociation World Congress*, vol. 1, pp. 55–60. Tsinghua University Press & Springer, Beijing 2005.

Konstrukce GLIF modelu a znalostní ontologie

doktorand:
ING. DAVID BUCHTELA

EuroMISE Centrum – Kardio
Ústav informatiky AV ČR
Pod Vodárenskou věží 2

182 07 Praha 8

buchtela@euromise.cz

školitel:
DOC. ING. ARNOŠT VESELÝ, CSC.

EuroMISE Centrum – Kardio
Ústav informatiky AV ČR
Pod Vodárenskou věží 2

182 07 Praha 8

vesely@euromise.cz

obor studia:
Informační management
číselné označení: 6209V

Abstrakt

Získané znalosti v různých oborech je možné reprezentovat ve formě oborových doporučení. Textová doporučení lze formalizovat grafickým GLIF modelem (GuideLine Interchange Format). Znalostní ontologie je chápána jako explicitní specifikace systému pojmů a zákonitostí modelující určitou část světa. Tento článek se zabývá možností použití znalostních ontologií ve fázi konstrukce GLIF modelu z oborových doporučení.

Klíčová slova: oborová doporučení, GLIF model, znalostní ontologie

1. ÚVOD

Znalosti získané v různých oborech je možné formalizovat do oborových doporučení usnadňujících rozhodování v daném konkrétním případě opírající se o znalostní bázi oboru. Oborová doporučení jsou obvykle distribuována v textové podobě, pro počítačovou implementaci a zpracování je však nutné mít tato doporučení ve strukturované formě. Proces formalizace textových doporučení do strukturované podoby není zcela triviální.

2. CÍL A METODIKA

Cílem tohoto příspěvku je porovnat způsoby reprezentace znalostí pomocí GLIF modelu a znalostních ontologií. Dále je v článku naznačena možnost použití ontologií při konstrukci GLIF modelu z textových oborových doporučení.

GLIF model

GLIF model (*GuideLine Interchange Format*), nejčastěji používaný pro přehlednou reprezentaci oborových doporučení, vznikl spoluprací Univerzity Columbia, Harvardské, McGillovy a Stanfordské univerzity. GLIF poskytuje objektově a procesně orientovaný pohled na oborová doporučení (viz [6] a [7]). Výsledným modelem je orientovaný graf skládající se z pěti hlavních částí (kroků):

- **akce** - představuje specifickou činnost nebo událost. Akcí může být i podgraf, který dále zjemňuje danou činnost.

- **rozhodování** - představuje větvení (výběr) na základě automatického splnění logického kritéria, kdy další postup grafem je dán výsledkem aritmetického nebo logického výrazu nad konkrétními daty, a nebo rozhodnutí uživatele, kdy akce reprezentovaná následným krokem může ale nemusí být provedena, nebo může probíhat paralelně s jinými akcemi. Uživatel má v tomto místě možnost se rozhodnout, kterou částí grafu bude dále pokračovat.

- **větvení a synchronizace** - větvení se používá při modelování nezávislých kroků, které mohou probíhat paralelně a synchronizace slouží pro tyto kroky jako slučovací bod.

- **stav** - značí stav, ve kterém se zkoumaný objekt nachází při vstupu do modelu nebo po provedení některého předchozího kroku.

Průchod jednotlivými větvemi GLIF modelu je závislý na splnění či nesplnění podmínek v rozhodovacích krocích (*podmínky typu strict-in a strict-out*) nebo na interaktivním výběru příslušné větve grafu uživatelem na základě doporučené či nedoporučené následné větve grafu (*podmínky typu rule-in a rule-out*).

GLIF model je grafický, pro další počítačové zpracování je proto nutné jej zakódovat ve vhodném formálním jazyku, např. GELLO [5] nebo XML [2]. GELLO (*Guideline Expression Language*) je objektově orientovaný dotazovací a vyjadřovací jazyk určený pro podporu rozhodování. XML (*eXtensible Markup Language*) je univerzální značkovací jazyk přímo popisující dané informace a jejich hierarchii. To je významné zejména při přenosu informací mezi různými systémy.

Ontologie

Uvádí se celkem sedm historicky vzniklých definic pojmu ontologie, které se do značné míry překrývají. Zde uvedeme pouze definici formulovanou T. Gruberem, jedním z duchovních otců ontologií: *"Ontologie je explicitní specifikace konceptualizace."* [4], a její modifikaci provedenou W. Borstem: *"Ontologie je formální specifikace sdílené konceptualizace"* [3]. Pojem konceptualizace znamená definování všech objektů, které uvažujeme při řešení úloh reálného světa (universum), vymezení relací a funkcionálních vztahů mezi nimi. Objekty universa mohou být konkrétní nebo abstraktní, reálně existující nebo fiktivní, jednoduché nebo složené.

Z hlediska znalostního inženýrství lze ontologie používané v procesu vývoje znalostní aplikace rovněž chápat jako znalostní modely, tedy abstraktní popisy (určité části) znalostního systému, které jsou relativně nezávislé na finální reprezentaci a implementaci znalostí. Podstatné je, že jde o modely sdílitelné více procesy v rámci jedné aplikace, a opakovaně použitelné pro různé aplikace, které mohou být oddělené časově, prostorově i personálně.

Podle předmětu formalizace lze ontologie rozdělit na následující typy [9]:

- *Doménové ontologie* jsou typem daleko nejfrekventovanějším. Jejich předmětem je vždy určitá specifická věcná oblast, vymezená šířeji (např. celá problematika medicíny nebo fungování firmy) či úžeji (problematika určité choroby, poskytování úvěru apod.).

- *Generické ontologie* usilují o zachycení obecných zákonitostí, které platí napříč věcnými oblastmi, např. problematiky času, vzájemné pozice objektů (topologie), skladby objektů z částí (mereologie) apod. Někdy se ještě výslovně vyčleňují tzv. ontologie vyšší úrovně, které usilují o zachycení nejobecnějších pojmů a vztahů, jako základu taxonomické struktury každé další (např. doménové) ontologie.

- *Jako úlohové ontologie* jsou někdy označovány generické modely znalostních úloh a metod jejich řešení. Na rozdíl od ostatních ontologií, které zachycují znalosti o světě (tak, jak je), se zaměřují na procesy odvozování. Mezi úlohy tradičně zachycené pomocí takových znalostních modelů patří např. diagnostika, zhodnocení, konfigurace, nebo plánování.

- *Aplikační ontologie* jsou nejspecifičtější. Jedná se o konglomerát modelů převzatých a adaptovaných pro konkrétní aplikaci, zahrnující zpravidla doménovou i úlohovou část (a tím automaticky i generickou část).

Základem znalostních ontologií jsou třídy, které označují množiny konkrétních objektů. Na rozdíl od tříd v objektově-orientovaných modelech a jazycích nezahrnují ontologické třídy procedurální metody. Jejich interpretace je spíše odvozena z pojmu relace v tom smyslu, že třída odpovídá unární relaci na dané doméně objektů. Na množině tříd bývá často definována hierarchie (taxonomie).

Individuum v ontologii odpovídá konkrétnímu objektu reálného světa, a je tak do jisté míry protipólem třídy. Termín instance je často chápán jako ekvivalentní, asociuje však příslušnost k určité třídě, což nemusí být nutně skutečností - individuum může být do ontologie provizorně vloženo i bez vazby na třídy.

Podobně jako v databázových modelech jsou podstatnou složkou ontologií vztahy čili relace n-tic objektů (individuí). Vedle výrazů explicitně vymezujících příslušnost ke třídám a relacím je obvykle možné do ontologií zařazovat další logické formule, vyjadřující např. ekvivalenci / subsumpci tříd či relací, disjunktnost tříd, rozklad třídy na podtřídy apod.. Nejčastěji se označují jako axiomy.

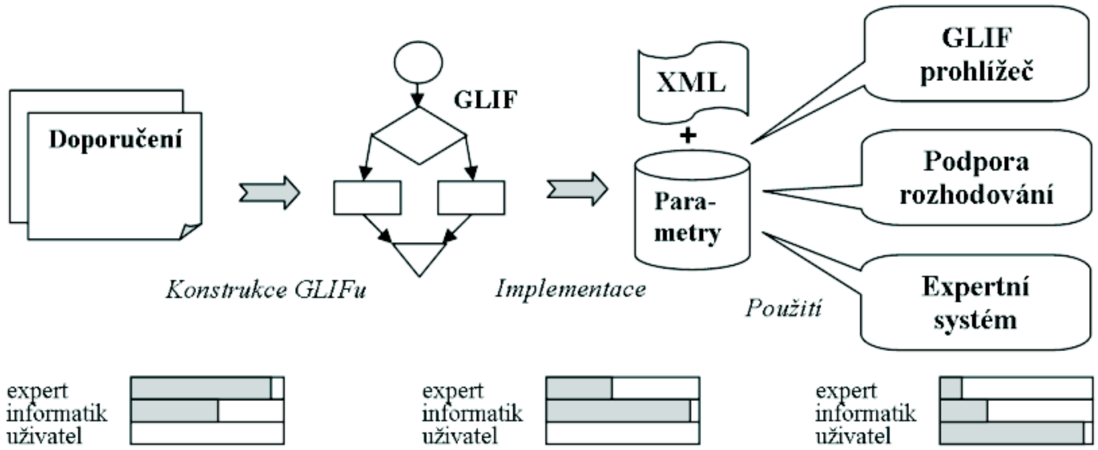
3. VÝSLEDKY A DISKUSE

Konstrukce GLIF modelu

Vytvoření GLIF modelu z textových doporučení a jeho implementace není zcela jednoduchou záležitostí a celý proces lze rozdělit na několik fází. Celý proces konstrukce a implementace je patrný z obrázku 1.

Ve fázi konstrukce GLIF modelu z textových doporučení je důležité najít procesní strukturu doporučení, všechny podstatné parametry modelu a jejich vzájemné vztahy. Základní parametry představují přímo měřitelné (nebo jinak získatelné) hodnoty, odvozené parametry se získají aritmetickou, logickou či logicko-aritmetickou operací nad základními parametry.

Při hledání procesní struktury doporučení, tj. stanovení algoritmu řešení daného problému, se jako nejefektivnější jeví vzájemná spolupráce informatika a experta z oboru (nejlépe autora příslušných textových oborových doporučení). Při tomto procesu lze s úspěchem využít technik návrhu informačních systémů na bázi metodologie UML (*Unified Modelling Languauge*) [8]



Obrázek 1: *Proces konstrukce a implementace GLIF modelu*

Při hledání parametrů a jejich vztahů lze využít některou z metod automatického dolování znalostí z textu [1]. Dokument lze reprezentovat vektorem termínů (slovo nebo víceslovné spojení), kde ke každému termínu je zjištěna frekvence výskytu daného termínu v dokumentu.

Tato reprezentace je velmi častá, skrývá v sobě však několik omezení:

- nepostihuje kontext, ve kterém se slovo objevilo
- je obtížné vyjádřit sémantickou podobnost termínů
- otázkou je, jak volit množinu reprezentativních termínů - velký počet termínů vede na extrémně velké a řídké matice frekvencí výskytu, malý počet naopak způsobuje ztrátu informace
- otázkou je i vhodná volba složitosti jednotlivých termínů
- problémem je i vlastní formulace doporučení, která jsou často určena k řešení speciálních úloh za předpokladu, že uživatel má určitou úroveň základních znalostí (neobsažených v textu)

Použití ontologií

V této fázi konstrukce by byly, za předpokladu jejich existence, významný pomocníkem doménové a generické ontologie vztahující se k problémové oblasti obsažené v oborových doporučeních. Znalosti jsou v ontologiích uloženy ve formě hierarchie (taxonomie) tříd pojmů a objektů reálného světa. Taková reprezentace umožňuje vyjádřit důležité pojmy modelované oblasti a jejich vzájemné vztahy.

Pomocí ontologií lze tedy vhodně určit počet a složitost termínů při dolování znalostí při maximálním omezení ztráty uložených informací. Ontologie napomáhají i modelování znalostí souvisejících znalostí přímo neobsažených v textu doporučení.

Další možností použití ontologií je implementace datového modelu, tj. uložení všech základních i odvozených parametrů GLIF modelu ve formě (ontologické) hierarchie tříd. Tento způsob uložení usnadňuje navázání parametrů GLIF modelu na reálná data uložená bázi dat týkající se konkrétního řešeného problému. Zároveň umožňuje sdílení parametrů mezi jednotlivými modely a aplikacemi.

Výsledkem fáze konstrukce je grafický GLIF model a datový model základních a odvozených parametrů, které co možná nejlépe odpovídají znalostem skutečně obsaženým v textových doporučeních.

4. ZÁVĚR

Modelování oborových doporučení v GLIF modelu není triviální proces a je nutné dodržet jisté podmínky, aby výsledný model co možná nejvíce odrážel znalosti obsažené v textové podobě oborových doporučení. V průběhu tohoto procesu hrají znalostní ontologie podstatnou roli, zejména při konstrukci datového modelu, tj. hledání všech podstatných parametrů GLIF modelu. Následně se uplatňují při jejich implementaci (uložení) vhodné pro snadné navázání na bázi dat konkrétního řešeného problému a sdílení s ostatními modely a aplikacemi.

Literatura

[1] Berka P., “Dobývání znalostí z databází”, *Academia, Praha*, 366 s., ISBN 80-200-1062-9, 2003.

[2] Buchtela D., “Implementace GLIF modelu v XML”, *Sborník příspěvků, PEF ČZU Praha*, str.48 + CD, ISBN 80-213-1150-9, 2004.

[3] Borst W.N., “Construction of Engineering Ontologies for Knowledge Sparing and Reuse”, *PhD dissertation, University of Twente, Enschede*,1997.

[4] Gruber T.R., “A Translation Approach to Portable Ontology Specifications”, *Knowledge Acquisition*,5(2), ISSN 1042-8143, 1993.

[5] Ogunyemi O., Zeng Q., Boxwala A.A., “BNF and built-in classes for object-oriented guideline expression language (GELLO)”, *Technical Report: Brigham and Women’s Hospital*, Report No.: DSG-TR-2001-018, 2001.

[6] Ohno-Machado L., Gennari J. H., Murphy S.N., Jain N.L., Tu S.W., Oliver D., et al., “The Guide-Line Interchange Format: A model for representing guidelines”, *Journal of the American Medical Informatics Association*, 5(4), pp. 357-372, ISSN 1067-5027, 1998.

[7] Peleg M., Boxwala A.A., et al., “GLIF3: a representation format for sharable computer-interpretable clinical practice guidelines”, *Journal of Biomedical Informatics*, Volume 37, Issue 3, pp. 147-161, ISSN 1532-0464, June 2004. 1067-5027, 1998.

[8] Schmuller J., “Myslíme v jazyku UML”, *Grada Publishing, Praha*, 360 s., ISBN 80-247-0029-8, 2001.

[9] Uschold M., Gruninger M., “Ontologies: principles, methods and applications.”, *The Knowledge Engineering Review*, Vol.11:2, pp. 93-136, ISSN 0269-8889, 1996.

Regression Benchmarking: An Approach to Quality Assurance in Performance

Post-Graduate Student:

ING. L. BULEJ, MGR. T. KALIBERA

Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2
182 07 Prague 8

Charles University in Prague
Faculty of Mathematics and Physics
Malostranské náměstí 25

118 00 Prague 1

lubomir.bulej@mff.cuni.cz
tomas.kalibera@mff.cuni.cz

Supervisor:

ING. PETR TŮMA, DR.

Faculty of Mathematics and Physics
Charles University in Prague
Malostranské náměstí 25
118 00 Prague 1
Czech Republic

petr.tuma@mff.cuni.cz

Field of Study:
Software systems
Classification: I-2

Abstract

The paper presents a short summary of our work in the area of regression benchmarking and its application to software development. Specially, we explain the concept of regression benchmarking as an application of classic benchmarking for the purpose of regression testing of software performance, the requirements for employing regression testing in a software project, and methods used for analyzing the vast amounts of data resulting from repeated benchmarking. We present the application of regression benchmarking on a real software project and conclude with a glimpse at the challenges for the future.

1. Introduction

Quality assurance in software can have many forms, ranging from testing of various high-level usage scenarios by human operators, to the low-level testing of basic functionality. With the increasing complexity of software, quality assurance has gained popularity and, in one form or another, slowly became a necessity for upholding the quality in large scale software projects, especially when the development is carried out by multiple developers or in distributed fashion.

While the human operators can be, to a certain degree, replaced by software robots simulating human behavior, the high-level testing is still rather costly and is typically carried out in commercial projects, where the cost of testing has its own place in the development budget. Low-level testing of functionality is usually much cheaper to come by. Low-level testing, called regression testing, can be performed automatically without human intervention and comprises a series of tests that are run on the software and are expected to return results that are known to be correct.

Even though the idea is simple, it has been very successful in discovering all kinds of programming errors. Obviously, this method is not suitable for discovering errors of the kind when the low-level functions return correct results but the software as a whole does not actually perform what was required of it. Such errors can be usually attributed to bad design and are best discovered by high-level testing.

Because of its simplicity and clearly visible benefits, regression testing has become a widely accepted practice in quality assurance and is for example an integral part of the Extreme Programming development paradigm. Probably the most prominent example of a framework which eases the integration of regression testing with a software project is JUnit [1], which makes it easy for developers to write tests for individual classes in their project.

Regression testing, as used today, typically exercises the code to find implementation errors which result in incorrect operation. This kind of testing usually neglects another aspect to software quality that would benefit from repeated testing, which is performance.

Performance of software often degrades with time, and the degradation is usually due to added features, sometimes due to subtle bugs that do not result in incorrect operation, and sometimes due to use of inappropriate algorithms for handling data structures. Without a performance track record, telling which of the modifications to the software code base caused the overall performance degradation is difficult when performance becomes an issue.

To that end, we have proposed to extend classic regression testing with regular testing of performance. The testing is carried out by benchmarking the key parts of the software, tracking the performance, and looking for changes in performance that resulted from modifications to the code base. Besides tracking the performance, which may serve to confirm expected performance improvements, the idea is to alert developers when a performance degradation is detected. If the degradation was unanticipated, locating the code modification which resulted in the degradation is easier when the degradation is reported soon after the modification was introduced into the code base.

In analogy to *regression testing*, this process is called *regression benchmarking*, being an application of benchmarking for the purpose of regression testing of software performance.

The rest of the paper is organized as follows: Section 2 describes the concept of regression benchmarking in greater detail and explains how it differs from classic benchmarking and what are the key requirements for employing regression benchmarking in quality assurance process. Section 3 briefly mentions the properties of the collected data and presents an overview of methods used for automatic processing of the data. Section 4 illustrates the application of regression benchmarking on a real world project and provides some directions for future work. Section 5 then concludes the paper.

2. More Than Just Benchmarking

There is much more to regression benchmarking than the term may suggest. When compared to classic benchmarking, we can observe several major differences, which stand behind the increased complexity of regression benchmarking.

Classic benchmarking is typically carried out to determine or evaluate performance of one or more software products. Such performance evaluation is only done occasionally and typically includes a human operator in the process. The collected data are processed into reports intended for human audience and the actual numbers are important for interpretation of the results. The measurements should follow the best practices used in the field and avoid pitfalls such as described in [2], but the precision of the measurements generally does not play a significant role as long as the measurement process remains transparent and takes place under identical circumstances, i.e. on the same hardware, operating system, etc.

Regression benchmarking, on the other hand, is performed regularly and the goal is to exclude any human intervention from the process. The subject of the benchmarking is always the same software project, but in multiple versions. The amount of collected data is vast and not intended for human audience – the data is processed and analyzed by a machine and developer attention is only required when a performance regression is found. Consequently, the absolute values of performance measures are not as important as relative changes in the values. Correct and robust benchmarking methodology is important to ensure reproducibility of the results in the same environment, which is especially difficult with today's hardware platforms and operating systems. An estimate of the precision of the results is important to improve the robustness of the machine analysis as well as to help determine the amount of data to collect to conserve computational resources.

2.1. Regression Benchmarking Requirements

We have first proposed the concept of regression benchmarking in [3, 4]. The feasibility of the concept for use with simple and complex middleware benchmarks has been demonstrated in [5, 6], yet the automatic analysis of results of complex benchmarks remains a challenge, mainly due to lack of robust automatic data clustering algorithms. In the course of above works, we have elaborated the requirements for incorporating regression benchmarking into development process of a software project, which can be split into following categories:

1. Benchmark Creation

   To conserve developer time, which is a valuable resource, the creation of new benchmarks for use in regression benchmarking must be easy and straightforward task. In our experience with middleware benchmarking, this is best achieved with a benchmarking suite, that provides the developer with portable, generic benchmarking facilities for precise timing, thread management, locking, etc., as well as domain-specific benchmarking facilities, e.g. for benchmarking CORBA middleware [7].

   If there is no benchmarking suite for a given domain, it should be fairly easy to adapt existing benchmarks for use with regression benchmarking, as we have done when setting up regression benchmarking project [8] for the Mono [9] project.

2. Benchmark Execution

   Benchmark execution must be fully automatic. This includes downloading (and possibly compiling), installing and executing all the software required regression benchmarking of a given software project. The execution of benchmarks includes non-intrusive monitoring of the running benchmarks and recovery from crashes, deadlocks and other kinds of malfunctions typical for software under development.

   In addition, all the activity associated with benchmark execution must be carefully orchestrated to avoid distortion of results and must observe a sound benchmarking methodology to provide robust and reproducible results.

   While this part of regression benchmarking may not be the grand challenge, it indeed is very complex and requires a great amount of highly technical work. Fortunately, if regression benchmarking is being set up for a specific project and using only simpler benchmarks, a lot of the complexity can be avoided.

3. Result Analysis

   Similar to benchmark execution, the analysis of benchmark results must be also fully automatic and represents a major challenge in regression benchmarking. Since the measurements are carried out on real systems, the measured data contain a significant amount of distorted values which, while not indicative of performance, cannot be simply filtered out [5]. Also, the complexity of today's hardware platforms and operating systems hinders reproducibility of benchmark experiments, which is most clearly visible on the problem of random initial state [10].

   Benchmark results are typically processed and analyzed using statistical methods, but there are no readily available statistical methods that would be designed to work with the kind of data produced by benchmarks. That is to say that the underlying distribution of the data is typically unknown, there is a significant amount of outliers present in the data, and that due to the random initial state, two identical benchmark experiments do not provide data that could be described by distributions with the same parameters.

As a result, automatic analysis of benchmark data requires a lot of processing to achieve robustness which is in turn required to minimize the amount of false alarms. More details on our current approach to data analysis are provided in Section 3.

2.2. Generic Benchmarking Environment

From the above list of requirements, it may seem that setting up an environment for regression benchmarking is not worth the effort. We believe though, that most of the requirements can be satisfied by a generic benchmarking environment which would relieve the developers of many of the tedious tasks and allow them them write their own benchmarks to be included in the process.

This approach is similar to that of JUnit, but is inevitably more complex because of the scope of the full automation requirement. In [11] we have elaborated the requirements and proposed a basic architecture of a generic benchmarking environment that is suitable for regression benchmarking. A project called BEEN [12] has been launched to implement the proposed environment.

There are several other projects that execute benchmarks regularly, most notably the TAO Performance Scoreboard [13] and the Real-Time Java Benchmarking Framework [14, 15] associated with the TAO [16] and OVM projects. Both projects benchmark their software daily using the above frameworks, but only only for the purpose of tracking performance. Both benchmarking frameworks are tailored to their associated software projects and their needs. Consequently, they do not provide the required scope of automation and functionality as required for regression benchmarking.

3. Sifting Through the Data

The automatic data analysis has two goals. The first, and primary, goal is to detect changes in performance. The second goal is to determine what amount of data is sufficient to detect the changes so that the computational resources are utilized effectively. There are several obstacles which make these goals difficult to achieve.

3.1. Using Statistics to Detect Changes

Even though computers are deterministic and the benchmarks repeatedly measure the duration of some operation, the data obtained in the measurements always show some, seemingly random, fluctuations. Figure 1 shows an example of data obtained from multiple executions of a FFT benchmark [17]. The horizontal axis bears an index of data sample while the vertical axis shows the time it took to perform the operation. Each execution of the benchmark collects a certain amount of data, multiple executions are separated by vertical lines.

As can be seen in Figure 1, the data in single run are dispersed around some central location. For the purpose of analysis, we consider the individual samples to be observations of multiple independent and identically distributed random variables. This reduces the analysis to the problem of finding the parameters of the underlying distribution of the data and the detection of changes to the problem of comparing whether data from two different versions of the software follow the same distribution.

The challenge remains in that the distribution the data should follow is unknown and that the data are not typically well behaved. Often they contain outliers, multiple clusters, and autodependency, if not between individual samples, then between clusters of samples.

We have shown that in some cases the standard statistical methods can detect changes in performance [5, 6] but unfortunately not in the general case, especially when we want to avoid human intervention during the analysis. For certain cases of misbehaving data, we can use some form of preprocessing or robust statistics to get useful results, as shown in [18, 19].

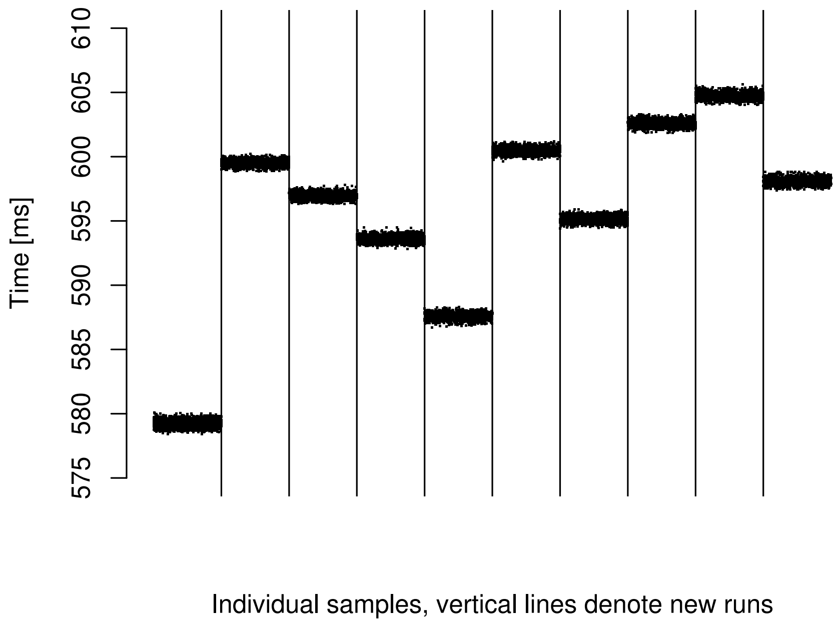


Figure 1: Data collected in multiple runs of FFT benchmark

3.2. The Problem with Precision

The detection of performance regressions must be highly reliable, because even a low number of false alarms may result in loss of trust from the developers. To reliably compare the performance of two versions of the same software, we need an estimate of precision of the values we are comparing.

Typically, the location and dispersion parameters of the distribution of the measured data are estimated as sample average and sample variance. The confidence interval for the sample average is then used as a measure of precision. However, this approach cannot be used even for the (relatively tame) data shown in Figure 1. The problem lies in that each execution of the same benchmark under the same circumstances results in a collection of samples for which the location parameter of their distribution shifts randomly with each execution.

More detailed description of the problem as well as method for determining the precision of a benchmark result using a hierarchical statistical model is provided in [10, 18, 19].

Depending on the type of benchmark and the tested software project, the precision of the result is more influenced either by the number of benchmark runs or by the number of samples collected in each benchmark run. Given a total cost of resources allotted to a benchmark experiment, we can determine the optimal number of samples that should be collected in each benchmark run so that the contribution of the samples to the precision of the result is maximized. The resulting precision then depends on the number of benchmark runs.

Knowing precision of the results, we can detect the changes in performance of the software project. While a difficult task in itself, it is certainly not the last challenge in regression benchmarking.

4. Pointing Out the Problems

While identifying changes is performance is difficult, there is more to be done to aid the developers in identifying the code modifications that caused them. We have set up regression benchmarking [8] for the Mono [9] project to serve as a testbed for our methods. The Mono project (virtual machine, run-time libraries) is an open source implementation of the .Net platform and we are using 5 different benchmarks to determine performance of daily development snapshots of the Mono environment.

Figure 2 shows the history of performance for the FFT benchmark taken from the SciMark C# [20] benchmarking suite. Each data point shows the confidence interval of the mean execution time and the lines connecting some of the data points indicate significant changes in performance, which were detected automatically by the regression benchmarking system.

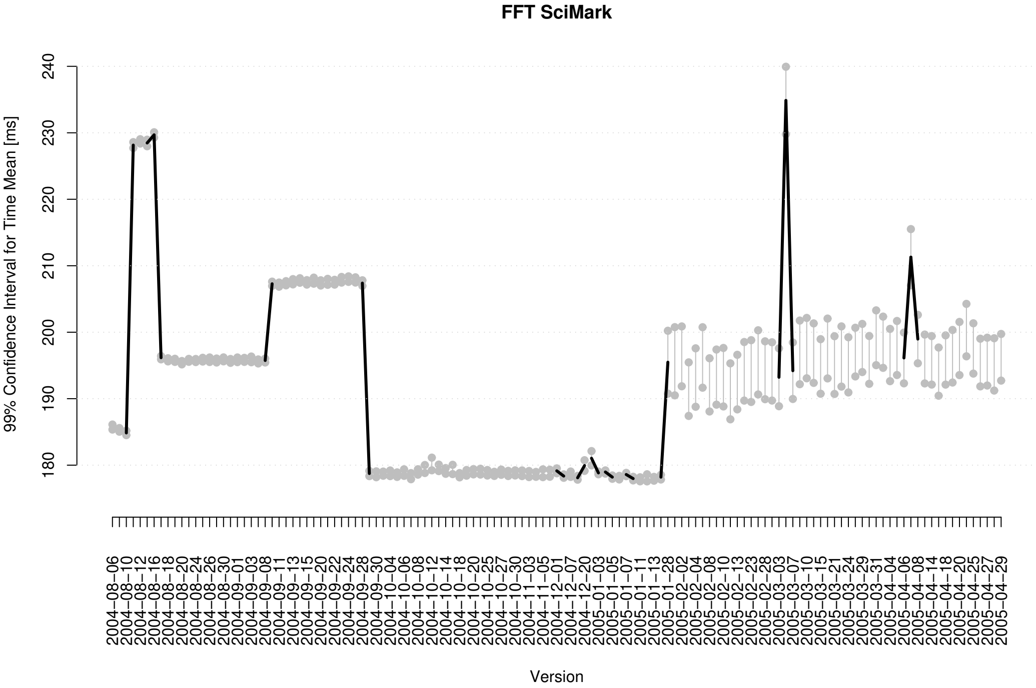


Figure 2: Data collected in multiple runs of FFT benchmark

The benchmarks are run daily and the results are immediately published on the web page of the project. To help the developers identify the code modifications causing the changes in performance, the regression benchmarking systems shows the differences between the two compared versions in the form of source code diff.

In [19] we have experimentally tracked some of the performance changes to the modifications in the code. The tracking can be much easier if the source code version control system groups logical changes to the project code. We realize that this may still require a lot of work from the developers, especially when the project is changing quickly.

The challenge for future work is to automate, at least partially, the finding of candidate source code modifications that may have caused particular change in performance, or at least excluding those that could not have caused it. Such modifications are mostly cosmetic changes in the source code, but it has been shown [21] that even this kind of changes may impact performance.

More advanced methods may attempt to correlate profiler data with the modifications for a particular version, or attempt to analyze portions of code affected by the changes and look for certain patterns that would suggest whether the change is purely cosmetic or whether it influences the behavior of the software.

5. Conclusion

Regression benchmarking is an approach to software quality assurance which aims to extend the widely accepted practice of regression testing with regular testing of performance.

Regression benchmarking employs classic benchmarking practice to obtain performance data which is then automatically processed and analyzed. The whole process of regression benchmarking needs to be fully automated to and request attention only when a performance regression has been found.

We have presented a summary of work done on various problems related to regression benchmarking, especially the automation of benchmark execution and statistical analysis of the performance data. The challenges for the future lie mostly with the automation of tracking performance regressions back to source code modifications.

The concept of regression benchmarking is being tested on an real-world project, the results of which are available on-line at [8]. So far the effort has resulted in identification of several performance regressions, which suggests that regression benchmarking can indeed contribute to the quality assurance process in software development.

Acknowledgments

This work was partially supported by the Grant Agency of the Czech Republic projects 201/03/0911 and 201/05/H014.

References

[1] JUnit Community, “Java unit testing.” http://www.junit.org.

[2] A. Buble, L. Bulej, and P. Tuma, “CORBA benchmarking: A course with hidden obstacles,” in *International Workshop on Performance Evaluation of Parallel and Distributed Systems, IPDPS 2003*, p. 279, IEEE Computer Society, 2003.

[3] L. Bulej and P. Tuma, “Current trends in middleware benchmarking,” in *Proceedings of the 2003 PhD Conference, ICS CAS*, Oct. 2003.

[4] L. Bulej and P. Tuma, “Regression benchmarking in middleware development,” in *Workshop on Middleware Benchmarking: Approaches, Results, Experiences, OOPSLA 2003*, Oct. 2003.

[5] L. Bulej, T. Kalibera, and P. Tuma, “Regression benchmarking with simple middleware benchmarks,” in *International Workshop on Middleware Performance, IPCCC 2004* (H. Hassanein, R. L. Olivier, G. G. Richard, and L. L. Wilson, eds.), pp. 771–776, IEEE Computer Society, 2004.

[6] L. Bulej, T. Kalibera, and P. Tuma, “Repeated results analysis for middleware regression benchmarking,” *Performance Evaluation*, vol. 60, pp. 345–358, May 2005.

[7] P. Tůma and A. Buble, “Open CORBA Benchmarking,” in *Proceedings of the 2001 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2001)*, SCS, Jul 2001.

[8] Distributed Systems Research Group, “Mono regression benchmarking project.” http://nenya.ms.mff.cuni.cz/projects/mono, 2005.

[9] Novell, Inc., “The Mono Project.” http://www.mono-project.com, 2005.

[10] T. Kalibera, L. Bulej, and P. Tuma, “Benchmark precision and random initial state,” in *Proceedings of the 2005 International Symposium on Performance Evaluation of Computer and Telecommunications*

Systems, July 24-28, 2005, Cherry Hill, New Jersey, USA (M. S. Obaidat, M. Marchese, J. Marzo, and F. Davoli, eds.), vol. 37 of *Simulation Series*, pp. 853–862, SCS, July 2005.

[11] T. Kalibera, L. Bulej, and P. Tuma, "Generic environment for full automation of benchmarking," in *SOQUA/TECOS* (S. Beydeda, V. Gruhn, J. Mayer, R. Reussner, and F. Schweiggert, eds.), vol. 58 of *LNI*, pp. 125–132, GI, 2004.

[12] Distributed Systems Research Group, "Benchmarking environment project." http://nenya.ms.mff.cuni.cz/been, 2005.

[13] Distributed Object Computing Group, "TAO performance scoreboard." http://www.dre.vanderbilt.edu/stats/performance.shtml, 2005.

[14] M. Prochazka, A. Madan, J. Vitek, and W. Liu, "RTJBench: A Real-Time Java Benchmarking Framework," in *Component And Middleware Performance Workshop, OOPSLA 2004*, Oct. 2004.

[15] "OVM predictability and benchmarking." http://www.ovmj.org/bench/, Nov. 2004.

[16] Distributed Object Computing Group, "TAO: The ACE ORB." http://www.cs.wustl.edu/∼schmidt/TAO.html, 2005.

[17] R. Mayer and O. Buneman, "FFT benchmark." ftp://ftp.nosc.mil/pub/aburto/fft.

[18] T. Kalibera, L. Bulej, and P. Tuma, "Automated detection of performance regressions: The Mono experience," in *(MASCOTS 2005)*, Sept. 2005.

[19] T. Kalibera, L. Bulej, and P. Tuma, "Quality assurance in performance: Evaluating Mono benchmark results," in *2nd International Workshop on Software Quality (SOQUA 2005)*, Lecture Notes in Computer Science, Springer, Sept. 2005.

[20] C. Re and W. Vogels, "SciMark – C#." http://rotor.cs.cornell.edu/SciMark/, 2004.

[21] D. Gu, C. Verbrugge, and E. Gagnon, "Code layout as a source of noise in JVM performance," in *Component And Middleware Performance Workshop, OOPSLA 2004*, Oct. 2004.

Automated Connector Synthesis for Heterogeneous Deployment

Post-Graduate Student:
RNDR. TOMÁŠ BUREŠ
Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2
182 07 Prague 8
bures@cs.cas.cz

Supervisor:
PROF. ING. FRANTIŠEK PLÁŠIL, DRSC.
Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2
182 07 Prague 8
plasil@cs.cas.cz

Field of Study:
Software Systems
Classification: I-2

This work was partially supported by the Grant Agency of the Czech Republic project 201/03/0911.

Abstract

Although component based engineering has alredy become a widely accepted parading, easy combining components from different component system in one application is still beyond possibility. In our long-term project we are trying to address this problem by extending OMG D&C based deployment. We rely on software connectors as special entities modeling and realizing component interactions. However, in order to benefit from connectors, we have to generate them automatically at deployment time with respect to high-level connection requirements. In this paper we show how to create such a connector generator for heterogeneous deployment.

1. Introduction and motivation

In the recent years, the component based programming became a widely accepted paradigm for building large-scale applications. There are a number of business (e.g., EJB, CCM, .NET) and academic (e.g., SOFA[12], Fractal[11], C2[10]) component models varying in maturity and features they provide. Although the basic idea of component based programming (i.e., composing applications from encapsulated components with well defined interfaces) is the same for all component systems, combining components for different component systems in one application is still beyond possibility. The main problems hindering this free composition of components comprise different deployment processes (e.g., different deployment tools), different ways of component interconnections (e.g., different middleware), and compatibility problems (e.g., differect component lifecycle and type incompatibilities). To allow for freely and uniformly composing, deploying, and running applications composed from components of different component systems (as depicted in Figure 1) is the aim of heterogeneous deployment.

In our recent work we have used OMG D&C [13] as a basis for unified deployment. OMG D&C defines a deployment of homogeneous application in a platform independent way. It describes processes and artifacts used in the deployment. OMG D&C follows the MDA paradigm; thus, the platform independent description of deployment is transformed to a particular platform specific model (e.g., deployment for CCM). In order to allow for the heterogeneous deployment, we have extended the OMG D&C model to make it capable of simultaneously handling components from different component systems [9], and we have introduced the concept of software connectors [3] to the OMG D&C model to take responsibility for component interactions [5].

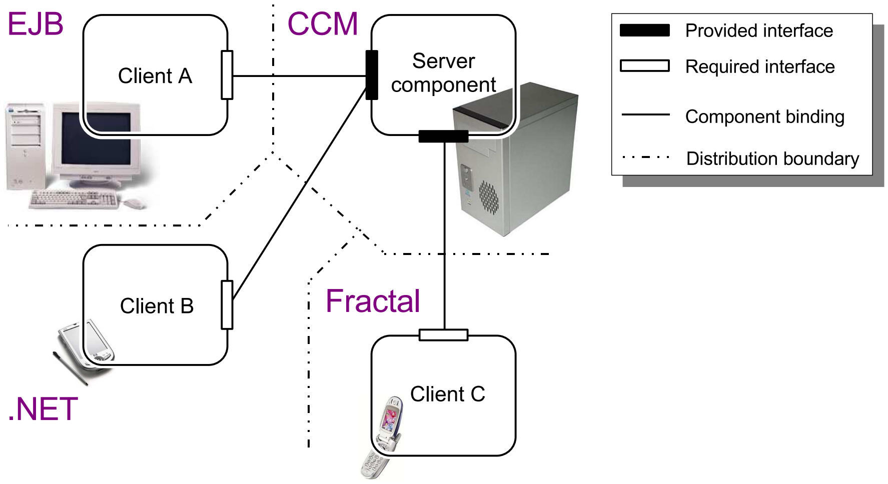


Figure 1: An example of heterogeneous component-based application

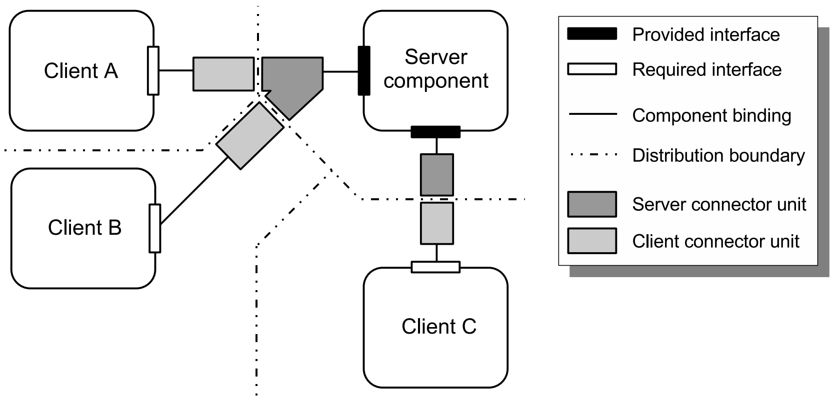


Figure 2: Using connector to interconnect components

Connectors being first class entities capturing communication among components (see Figure 2) help us at design time, to formally model intercomponent communication, and at runtime, to implement the communication. Connectors seamlessly address distribution (e.g. using a particular middleware). They also provide a perfect place for adaptation (for overcoming incompatibilities resulting from different component systems) and value added services (e.g. monitoring). Important feature of connectors is that an implementation of a particular connector is prepared as late as at deployment time, which allows us to tailor the connector implementation to specifics of a particular deployment node.

The fact that connectors are prepared at deployment time, however, brings a problem of their generation. They cannot be provided in a final form in advance. Forcing a developer to be present at deployment stage to tailor connectors to a particular component application and deployment nodes is not feasible either. Our solution to this problem is to benefit from the domain specificity of connectors and to create a connector generator which would synthesize connectors automatically (i.e., without human assistance) with respect to a high-level connector specification. As the specification we take connection requirements, which are associated with bindings in OMG D&C component architecture (e.g., a requirement stating that a connection should be secure with a key no weaker than 128-bits). In our case the the connection requirements are expressed as a set of name-value pairs (e.g., $minimal\_key\_length \rightarrow 128$). Also, the connector synthesis reflects particular target deployment environment (e.g., if both the components connected by the binding required to be secure reside on one computer, then the security is assure even without encryption). The description of the environment is taken from OMG D&C target data model.

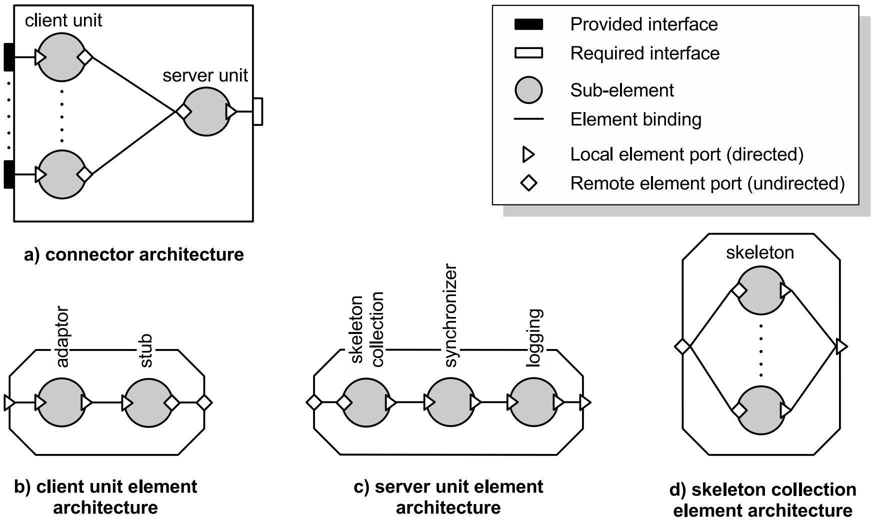


Figure 3: A sample connector architecture

2. Goals and structure of the text

In this work we present our approach to automated connector synthesis for heterogeneous deployment. We base the work on our previous experiences with building a connector generator for SOFA component system [4]. However, the connector generation in SOFA was not fully automatic — a connector structure must have been precisely specified — and it did not allow for handling and combining different component systems (i.e., it was homogeneous).

In this paper, we show how to create a connector generator that allows for the heterogeneity and works fully automatically, generating connectors with respect to target environment and higher-level connection requirements.

The paper is structured as follows. Section 3 shows the connector model we use. Section 4 describes the architecture of the connector generator and discusses certain parts of it in greater detail. Section 5 presents the related work and Section 6 concludes the paper.

3. Connector model

Software connectors are first class entities capturing communication among components. They are typically formalized by a connector model, which is a way of modeling their features and describing their structure. The connector model thus allows us to break the complexity of a connector to smaller and better manageable pieces, which is vital for automated connector generation. In our work we have adopted (with modifications) the connector model used in SOFA, since it has been specially designed to allow for connector feature modeling and code generation. In the rest of this section, we briefly describe this model along with the modification we have made to it.

Our connector model is based on component paradigm (see Figure 3. A connector is modeled as composed of *connector elements*. An element is specified by *element type*, which on the basic level expresses the purpose of the element and *ports*. Ports play the role of element interfaces. We distinguish three basic type of ports: a) provided ports, b) required ports, and c) remote ports. Elements in a connectors are connected via bindings between pairs of ports. Based on the type of ports connected we distinguish between a *local binding* (between required-provided or provided-required ports) and a *remote binding* (between two remote ports). Local bindings are realized via a local (i.e., inside one address space) call. Remote bindings are realized via a particular middleware (e.g., RMI, JMS, RTP, etc.). Since it is not possible to easily capture

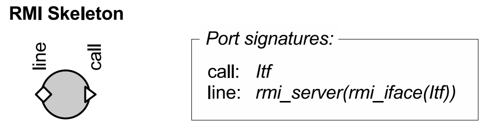


Figure 4: Specification of formal signatures of ports on an element.

the direction of the data flow on a remote binding (i.e., who acts as a client and who acts as a server) we view these bindings as undirected (or bidirectional).

On the top-level a connector is captured by a *connector architecture*, which defines the first level of nesting. All inter-element bindings in a connector architecture are remote. Thus, elements in a connector architecture encapsulate all the code which runs in one particular address space. Alternatively, we call these elements on the first level of nesting *connector units*.

Elements in an architecture are implemented using *primitive elements* and *composite elements*. A primitive element is just code (or rather a code template as we will se later). Apart from code, a composite element consists also of an *element architecture*. Binding between sub-elements in an element architecture may be only local. Thus, an element cannot be split among more address spaces.

Figure 3 gives an example of a connector realizing a remote procedure call. The top-level connector architecture is in Figure 3a. It divides a connector to one server unit and a number of client units. An implementation of the client unit is given in Figure 3b. It implements the client unit as a composite element consisting of an adaptor (which is an element realizing simple adaptations in order to overcome minor incompatibility problems) and a stub. Notice that the stub exposes on the left-hand side a provided interface (i.e., local) and on the right hand side it exposes a remote interface, which is use to communicate with a skeleton on the other side using a particular middleware. An implementation of the server unit (in Figure 3c) comprises a logging element (which is responsible for call monitoring), a synchronizer (which is an element realizing a particular threading model), and element called skeleton collection. The architecture of the skeleton collection element is given in Figure 3d. Its purpose is to group a number of skeletons implementing different middleware protocols, and thus to allow the connector to support different middleware on the server side.

A concrete connector is thus built by selecting a connector architecture and recursively assigning element implementations (either primitive or composite). The resulting prescription stating what connector architecture and what element implementations is called a *connector configuration*.

Connectors are in a sense entities very similar to components, however, there are a few basic differences between components and connectors: a) Connectors are typically span different address spaces and deal with middleware, while component (at least the primitive ones) reside in one address space only. b) Connectors have a different lifecycle to components — remote links between connector units have to be established before a connector can be put between components, and connectors may appear at runtime as component references are passed around. c) Connectors are in fact templates, which are later adapted to a particular component interface (as opposed to components which have fixed business interfaces).

In our model we do not only view connectors as whole as templates, but we view also connector elements as templates. Thus, an adaptation of a connector implies adaptation of all elements inside a connector[1]. As elements are templates, their signatures are variable, typically with some restrictions and relations to other ports of an element. To capture the restrictions and relations, we use interface variables and functions. An example is given if Figure 4. The skeleton element has two ports $call$ and $line$. The RMI implementation of the skeleton element as depicted in the example prescribes no restriction on the $call$-port. The actual signature[2] of the $call$-port is assigned to the interface variable Itf. The formal signature of the $line$-port is

[1]Although this approach is more tedious, it allows us to perform static invocation inside the connector; as opposed to having elements with general interfaces and being forced to perform dynamic invocation. The main benefit of static invocation over the dynamic one is better performance.

[2]When necessary we distinguish in this text between *formal port signature* and *actual port signature*. By formal signatures we

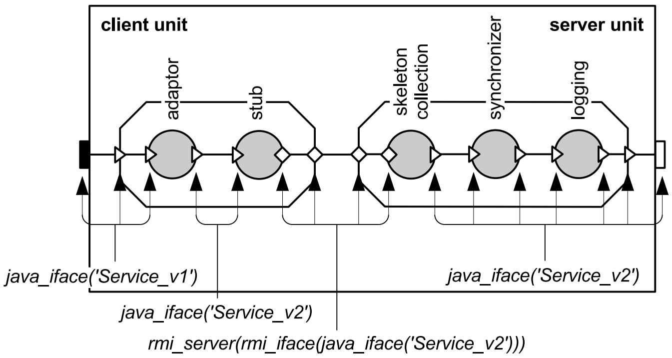


Figure 5: An example of interface propagation thorough a connector.

more complicated. It expresses the fact the original interface Itf is accessible via RMI. It uses the variable Itf to refer to the actual signature of the $call$-port and two functions rmi_server and rmi_iface. The rmi_iface function changes the interface Itf by adding neccessary features so that the interface can be used by RMI — it modifies the interface to extend `java.rmi.Remote` and changes the signature of every method in the interface to throws `java.rmi.RemoteException`. The rmi_server function expresses that the interface it takes as a parameter is accessible remotely via RMI. Notice that we do not use the functions only to capture changes in the interface itself (e.g., rmi_iface), but also to assign a semantics (e.g., rmi_server).

To perform the adaptation of a connector to a particular component interface we need to know the actual signature of each element port of every element inside the connector. To realize this, we assign the adjoining components' interfaces to respective interfaces of the connector and let the interface propagate through the elements inside the connector. The result of this process is shown in Figure 5. The $'Service\_v1'$ and $'Service\_v2'$ in the example are names of a sample component interfaces; the $java\_iface$ function returns an interface identified by a name. The adaptor element in the example solves incompatibilities between the two interfaces.

4. Connector generator

The connector model we have presented in the previous section very clearly outlines how connectors are generated — we synthesize a connector configuration and we assemble and adapt a connector accordingly. From the implementation point of view this is, however, more complicated.

An important fact to note is that we are not actually interested in building a connector as whole. Rather we want to build a server part and different client parts separately, as it better corresponds to the lifecycle and evolution of an application. Thus, in the rest of the text we focus on generation of connector units.

We have designed the connector generator from a few interoperating components (see Figure 6). The *generation manager* orchestrates the connector generation. The *architecture resolver* is responsible for creating a connector configuration with respect to high-level connection requirements and a target deployment environment (as described in Section 1). The *element generator* adapts element templates to particular interfaces and creates builder code for composite element architectures. Since we are interested only in connector units which are modeled also as elements, we can perform all the code generation only with help of the element generator. The *connector template repository* holds connector architectures and element implementations (i.e., code templates and composite element architectures). It is used by the architecture resolver to create a connector configuration and by the element generator to retrieve code templates to be adapted. The *con-*

mean the interface variables and restrictions as declared in an element implementation represented as a template. By actual signatures we mean the interfaces as values which are assigned to element ports and to which the element implementation is adapted. The terminology is in some sense similar to *formal* and *actual arguments* known from programming languages.

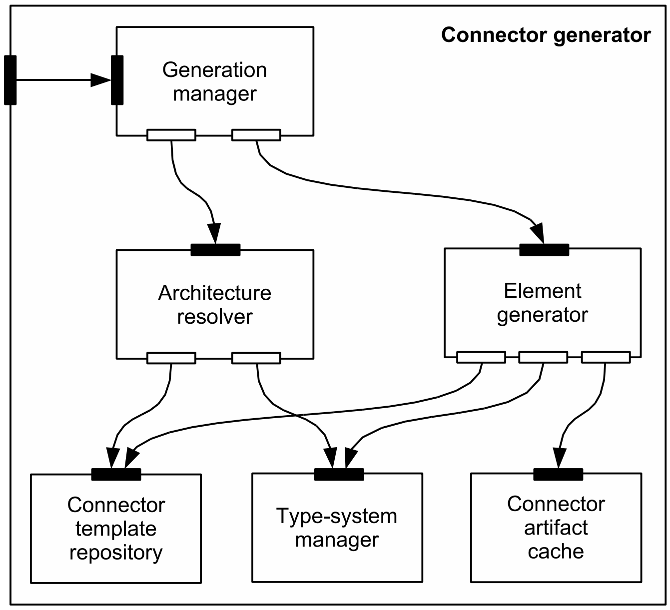


Figure 6: The architecture of the connector generator.

nector artifact cache is a repository where adapted elements are stored and from which they can be reused when needed next time. It addresses the problem that the code generation tends to be slow. Finally, the *type system manager* is a responsible for providing unified access to type-information originating from different sources and being in different format.

The synthesis of a connector configuration performed in the architecture resolver is discussed in more detail in Section 4.1. The element generator is elaborated on in Section 4.2. Due to the space constraints we have omited the details on the handling different type-systems, which, however, can be found in [8].

4.1. Architecture Resolver

We have implemented the architecture resolver in Prolog. We fill the Prolog knowledge base with predicates capturing information about architectures and elements kept in the connector template repository. Then we use the inherent backtracking in Prolog to traverse the search tree of various connector configurations and to find the configuration satisfying our requirements.

For every element implementation in the connector template repository we introduce to the knowledge base a predicate `elem_desc` (see Figure 7). The purpose of the predicate is to build a structure representing an instance of the element in a connector configuration (we call the structure *resolved element*). The resolved element contains all information required by the element generator (i.e., the name of the element implementation, which is a reference to the element's code template, and actual signatures of ports). In the case of a composite element, the resolved element includes also resolved elements of the sub-elements. Thus, the connector configuration in our approach has the form of a set of resolved elements for the units of the connector.

The auxiliary predicates used in the `elem_desc` predicate are in charge of constructing the resolved element. Predicate `elem_decl/3` creates its basic skeleton, predicate `elem_inst/7` chooses and constructs a sub-element, predicates `provided_port/2`, `remote_port/2` create the element ports, and predicates `binding/4` and `binding/5` create delegation and inter-element bindings respectively.

The `elem_desc` predicates are built based on the information kept in repository. To partially abstract from the Prolog language and to capture the element description in more comprehensible way, we use an XML notation which is during resolving transformed to the Prolog predicates. An example of the XML element

```
elem_desc(logged_client_unit,rpc_client_unit,Dock,This,CurrentCost,NewCost) :-
  elem_decl(This,logged_client_unit,rpc_client_unit),
  cost_incr(CurrentCost,0,CurrentCost0),
  elem_inst(This,Dock,SE_stub,stub,stub,CurrentCost0,CurrentCost1),
  elem_inst(This,Dock,SE_logger,logger,logger,CurrentCost1,NewCost),
  provided_port(This,call),
  remote_port(This,line),
  binding(This,call,logger,in),
  binding(This,logger,out,stub,call),
  binding(This,line,stub,line).
```

Figure 7: Prolog predicate describing a sample client_unit implementation

description is shown in Figure 8. The XML description comprises also parts related to code generation which are use by the element generator. As these are not important for us now, we have omited them in the example (marked by the three dots).

```
<element name="logged_client_unit" type="client_unit" ... >

  <architecture cost="0">
    <inst name="logger" type="logger"/>
    <inst name="stub" type="stub"/>
    <binding port1="call" element2="logger" port2="in"/>
    <binding element1="logger" port1="out" element2="stub" port2="call"/>
    <binding element1="stub" port1="line" port2="line"/>
  </architecture>

  ...

</element>
```

Figure 8: XML specification of a sample client_unit implementation — architectural part

Using the `elem_desc` predicates we are able to build a connector configuration, however, what remains to ensure is that the connector configuration specifies a working connector which respects the connection requirements. We have formulated these concerns as three consistency requirements — a) cooperation consistency, b) environment consistency, and c) connection requirements consistency. They are reflected in the following way.

Cooperation consistency. It assures that elements in a connector configuration can "understand" one another. We realize this by unifying signatures on two adjoining ports (in predicate `binding`). Recall that the signatures encode not only the actual type but also semantics (e.g., $rmi\_server(rmi\_iface('CompIface'))$).

Environment consistency. It assures that the element can work in a target environment (i.e., it requires no library which is not present there, etc.). We realize this by testing requirements on environment in `elem_desc` predicate (the description of environment capabilities is in the variable `Dock`).

Connection requirements consistency. It assures that the connector configuration reflects all connection requirements. Since some connection requirements require the whole connector configuration to be verifiable, we do not check them on-the-fly during building the connector configuration, but we verify them once the configuration is complete. We realize the checking by predicates able to test that a given connection requirement (in our case expressed as a name-value property) is satisfied by a particular connector configuration. In order to not loose extensibility and maintainability, we compose the predicate from a number of predicates verifying the connection requirement for a particular element implementation. In the case of a primitive element the connection requirement is verified directly. In the case of a composite element, the verification is typically delegated to a sub-element responsible for the functionality related to the connection requirement. If there is no predicate for the connection requirement associated with the element, the verification automatically fails. The example in Figure 9 shows the predicates for checking a "logging" property for a primitive element (console_log in our example) and for a composite element (logged_client_unit in our example).

```
nfp_mapping(This,logging,console) :-
  This = element(console_log,_,_,_,_).

nfp_mapping(This,logging,Value) :-
  This = element(logged_client_unit,_,_,_,_),
  get_elem(This,logger,SE_Logger),
  nfp_mapping(SE_Logger,logging,Value).
```

Figure 9: An example of predicates verifying connection requirements for a primitive and a composite element

To select the best configuration we use a simple cost function. We assign a weight to every element implementation reflecting its complexity. The cost of a connector configuration is then computed as a sum of weights of all element implementations in a configuration. We then select the configuration that has the minimal cost. To prune too expensive configurations on-the-fly already during their building, we use the `cost_incr/2` predicate (as shown in Figure 7).

4.2. Code generation

The connector configuration (which is the result of the process described in the previous section) provides us with a selection of element implementations and prescription to which interfaces they should be adapted. The next step in the connector generation is the actual adaptation of elements and generation of so called *element builder* (i.e., a code artifact which instantiates and links elements according to an element architecture) for each composite element.

In our approach, we generate source code and compile it to binary form (e.g., Java bytecode). The exact process of element's code generation is captured by element's specification. An example of such specification is shown in Figure 10. The omited parts correspond to specification of element's architecture and signatures previously shown in Figure 8.

The code generation is specified as a script of actions that have to be performed. In our example the action *jimpl* calls the class `CompositeGenerator`, which creates the source code (i.e., a Java class `LoggedClientUnit`) based on the actual signature it accepts as input parameters. The `CompositeGenerator` works actually as a template expander providing content for tags used in a static part of element's code template (in our case stored in file `compound_default.template` — see Figure 11). By dividing the code template to the static and the dynamic part (the template expander), we can reuse one template expander for a number of elements. Moreover, by using inheritance to implement the template expanders and providing abstract base classes for common cases we can keep the amount of work needed to implement a new code template reasonably small.

5. Related work

To our knowledge there is no work directly related to our approach, however, there are a number of projects partially related at least in some aspects.

Middleware bridging. An industry solution to connect different component models (or rather middleware they use) is to employ middleware bridges (e.g. BEA WebLogic, Borland Janeva, IONA Orbix and Artix, Intrinsyc J-Integra, ObjectWeb DotNetJ, etc.). A middleware bridge is usually a component delegating calls between different component models using different middleware. In this sense, it acts as a special kind of connector. However, middleware bridges are predominantly proprietary and closed solution, allowing to connect just two component models for which a particular bridge was developed and often working in one direction only (e.g. calling .NET from Java).

```
<element name="logged_client_unit" ... impl-class="LoggedClientUnit">

  ...

  <script>
    <command action="jimpl">
        <param name="generator" value="org.objectweb.dsrg.deployment.
        connector.generator.eadaptor.elements.generators.CompositeGenerator"/>
        <param name="class" value="LoggedClientUnit"/>
        <param name="template" value="compound_default.template" />
    </command>

    <command action="javac">
        <param name="class" value="LoggedClientUnit"/>
    </command>

    <command action="delete">
        <param name="source" value="LoggedClientUnit"/>
    </command>

  </script>

</element>
```

Figure 10: XML specification of a sample client_unit implementation — code generation part

```
package %PACKAGE%;

import org.objectweb.dsrg.deployment.connector.runtime.*;

public class %CLASS% implements ElementLocalServer, ElementLocalClient,
    ElementRemoteServer, ElementRemoteClient {

  protected Element[] subElements;
  protected UnitReferenceBundle[] boundedToRemoteRef;

  public %CLASS%() {
  }

%INIT_METHODS%

}
```

Figure 11: A static part of element's code template — file `compound_default.template`

Modelling connectors. A vast amount of research has been done in this area. There are a number of approaches how to describe and model connectors (e.g., [1], [14], [7]). Save the [1] which is able to synthesize mediators assuring that a resulting system is dead-lock free, the mentioned approaches are not generally concerned with code generation. That is why we have used our own connector model, which has been specifically designed to allow for code generation. Also, we are rather interested in rich functionality than formal proving that a connector has specific properties; thus, at this point we do not associate any formal behavior with a connector.

Configurable middleware. In fact connectors provide configurable communication layer. From this point of view, the reflective middleware (e.g., OpenORB [2]) being built from components is a very similar approach to our. The main distinction between the reflective middleware and connectors is the level of abstraction. While the reflective middleware deals with low-level communication services and provides a middleware

API, connectors stand above the middleware layer. They aim at interconnecting components in a unified way and for this task transparently use and configure a middleware (e.g., OpenORB).

6. Conclusion and future work

In this paper we have shown how to synthesize connectors for the OMG D&C-based heterogeneous deployment. We have used our experience from building the connector generator for SOFA. However, compared to SOFA, where the generator was only semi-automatic and homogeneous, we had to cope with the heterogeneity (which caused the co-existence of different type-systems) and fully automatic generation based on target environment and higher-level connection requirements.

We have created a prototype implementation of the connector generator in Java with an embedded Prolog [6] for resolving connector configurations. Currently we are integrating the generator with other our tools for heterogeneous deployment. As for the future work we would like to concentrate on automatic handling of incompatibilities between different component systems (i.e., life-cycle incompatibilities and type incompatibilities).

References

[1] M. Autili, P. Inverardi, M. Tivoli, D. Garlan, "Synthesis of "correct" adaptors for protocol enhancement in component based systems", Proceedings of SAVCBS'04 Workshop at FSE 2004. Newport Beach, USA, Oct 2004

[2] G. S. Blair, G. Coulson, P. Grace, "Research directions in reflective middleware: the Lancaster experience" Proceedings of the 3rd Workshop on Adaptive and Reflective Middleware, Toronto, Ontario, Canada, Oct 2004.

[3] D. Bálek and F. Plášil, "Software Connectors and Their Role in Component Deployment", Proceedings of DAIS'01, Krakow, Kluwer, Sep 2001

[4] L. Bulej and T. Bureš, "A Connector Model Suitable for Automatic Generation of Connectors", Tech. Report No. 2003/1, Dep. of SW Engineering, Charles University, Prague, Jan 2003

[5] L. Bulej and T. Bureš, "Using Connectors for Deployment of Heterogeneous Applications in the Context of OMG D&C Specification", accepted for publication in proceedings of the INTEROP-ESA 2005 conference, Geneva, Switzerland, Feb 2005

[6] E. Denti, A. Omicini, A. Ricci, "tuProlog: A Light-weight Prolog for Internet Applications and Infrastructures", Practical Aspects of Declarative Languages, 3rd International Symposium (PADL'01), Las Vegas, NV, USA, Mar 2001, Proceedings. LNCS 1990, Springer-Verlag, 2001.

[7] J. L. Fiadeiro, A. Lopes, M. Wermelinger, "A Mathematical Semantics for Architectural Connectors", In Generic Programming, pp. 178-221, LNCS 2793, Springer-Verlag, 2003

[8] O. Gálik and T. Bureš, "Generating Connectors for Heterogeneous Deployment", accepted for publication in proceedings of the SEM 2005 workshop, Jul 2005

[9] P. Hnětynka, "Making Deployment of Distributed Component-based Software Unified", Proceedings of CSSE 2004 (part of ASE 2004), Linz, Austria, Austrian Computer Society, ISBN 3-85403-180-7, pp. 157-161, Sep 2004

[10] N. Medvidovic, N. Mehta, M. Mikic-Rakic, "A Family of Software Architecture Implementation Frameworks", Proceedings of the 3rd WICSA conference, 2002

[11] ObjectWeb Consortium, Fractal Component Model, http://fractal.objectweb.org, 2004

[12] ObjectWeb Consortium, SOFA Component Model, http://sofa.objectweb.org, 2004

[13] Object Management Group, "Deployment and Configuration of Component-based Distributed Applications Specification", http://www.omg.org/docs/ptc/03-07-02.pdf, Jun 2003

[14] B. Spitznagel and D. Garlan, "A Compositional Formalization of Connector Wrappers", Proceedings of ICSE'03, Portland, USA, May 2003

Změkčování hran jako úloha strojového učení

doktorand:
MGR. JAKUB DVOŘÁK

Ústav informatiky AV ČR
Pod Vodárenskou věží 2

182 07 Praha 8

dvorak@cs.cas.cz

školitel:
RNDR. PETR SAVICKÝ, CSC.

Ústav informatiky AV ČR
Pod Vodárenskou věží 2

182 07 Praha 8

savicky@cs.cas.cz

obor studia:
Teoretická informatika
číselné označení: I1

Abstrakt

Změkčování hran je technika strojového učení sloužící ke zlepšení predikce metod založených na rozhodovacích stromech. Hledání takového změkčení hran v daném rozhodovacím stromu, které dává nejlepší (či dostatečně dobré) výsledky na trénovací množině, se ukazuje jako dosti obtížná optimalizační úloha. Tento článek se zabývá základními aspekty takovéto úlohy, které je třeba uvážit při hledání vhodného algoritmu či heuristiky k jejímu řešení. Také je ukázáno na výsledcích experimentů, jak může technika změkčování hran zlepšit predikci klasifikátoru a že tedy zkoumání této problematiky má praktický význam.

1. Úvod

Tradiční metody strojového učení založené na rozhodovacích stromech, např. CART [3], C4.5 [5] či C5.0, generují stromy, v nichž je každému vnitřnímu uzlu v_j přiřazena podmínka tvaru $x_{a_j} < c_j$, kde x_{a_j} je nějaký atribut předloženého vzoru a c_j je konstanta specifická pro tento uzel. [1]

Princip změkčování hran spočívá v tom, že je-li hodnota testovaného atributu x_{a_j} blízko rozdělující hodnotě c_j, potom se postupuje oběma větvemi vedoucími z uzlu v_j a výsledná predikce je kombinací takto získaných predikcí s vahami závisejícími na vzdálenosti x_{a_j} od c_j. Tento postup byl použit již v metodě C4.5 [5], kde motivací byla existence úloh, u nichž ostré (nezměkčené) rozhodování neodpovídá expertní znalosti problematiky — například v problému rozhodování, zda klientovi banky vydat kreditní kartu na základě aktuálního zůstatku na jeho účtu, jeho pravidelného příjmu a dalších podobných atributů, kde mírná nedostatečnost v některém z ukazatelů může být vyvážena výraznou saturací v jiných.

Další motivací ke změkčování hran je známý problém tradičních metod strojového učení založených na rozhodovacích stromech, že strom získaný takovou metodou určuje rozdělení vstupního prostoru úlohy pomocí nadrovin kolmých na osy prostoru, čímž vzniknou hyperkvádry (jež mohou být v některých směrech nekonečné) takové, že pro všechny body v jednom hyperkvádru strom predikuje stejnou výstupní hodnotu. Strom se změkčenými hranami umožňuje i složitější tvary predikce. Přitom zůstává do značné míry zachována výhoda přímočaré interpretace rozhodovacího stromu pro lidského experta, pouze jednoznačná rozhodovací pravidla jsou nahrazena fuzzy-pravidly.

[1] Ve stromu ve skutečnosti mohou být i uzly, v nichž se rozhoduje podle nominálního atributu, kde přiřazená podmínka nemá tento tvar. V těchto uzlech se změkčování neprovádí, proto o nich nadále neuvažujeme. To však nijak neomezuje použitelnost techniky změkčování hran, protože v praxi lze ve stromu snadno kombinovat uzly se změkčením s uzly s jinými rozhodovacími podmínkami při zachování platnosti následujících úvah.

Budeme se zde zabývat změkčováním hran v rozhodovacích stromech jakožto metodou postprocessingu stromů — tedy předpokládáme, že máme daný rozhodovací strom získaný některou z výše zmíněných tradičních metod, jakožto aproximaci neznámé funkce u, dále máme k dispozici trénovací data, tedy množinu vzorů z definičního oboru funkce u, z nichž u každého je známá funkční hodnota funkce u. Hledáme nějaké změkčení hran, které dává co nejlepší aproximaci funkce u.

V dalším textu se zaměříme na změkčování hran ve stromech pro klasifikaci, ačkoliv některé ze závěrů mají platnost i v případě regrese.

2. Parametry změkčení

Abychom mohli změkčování hran formulovat jako úlohu strojového učení, je třeba určit množinu možných změkčení resp. vymezit, mezi jakými změkčeními budeme hledat to nejlépe vyhovující. K tomu nejprve formálně upřesníme, jak pomocí rozhodovacího stromu se změkčenými hranami klasifikujeme předložený vzor.

Z vnitřního uzlu v_j, kterému je v původním (nezměkčeném) stromu přiřazena podmínka

$$x_{a_j} \leq c_j \tag{1}$$

vycházejí dvě hrany, z nichž označíme $e_{j,left}$ tu, po které se postupovalo při splněné podmínce (1) a $e_{j,right}$ druhou. Každé takové dvojici hran vycházejících z jednoho uzlu přiřadíme dvojici změkčujících křivek, tj. funkcí

$$f_{j,left}(x), f_{j,right}(x) : \mathbb{R} \to \langle 0, 1\rangle$$

takových, že jejich součet je 1. Hodnoty těchto funkcí v bodě x_{a_j} (hodnota příslušného atributu předloženého vzoru) určují váhy, s jakými se započítávají predikce příslušných větví stromu. Proto další rozumné požadavky na tyto funkce jsou, aby

$$f_{j,left}(c_j) = f_{j,right}(c_j) = \frac{1}{2}$$

a dále aby $f_{j,left}(x)$ pro $x \ll c_j$ bylo rovno nebo aspoň blízko 1 a pro $x \gg c_j$ bylo rovno nebo blízko 0. Tento požadavek vyjadřuje, že změkčování se týká jen vzorů, které jsou blízko rozdělující hodnotě c_j z (1).

Když listy stromu poskytují odhad pravděpodobností příslušnosti předloženého vzoru do jednotlivých tříd jako hodnoty $p(b|l)$ pro list v_l a třídu s indexem b, potom odhad pravděpodobností příslušnosti předloženého vzoru $\mathbf{x}$ daný stromem se změkčenými hranami lze vyjádřit:

$$P(b|\mathbf{x}) = \sum_{v_l \in Leaves} p(b|l) \prod_{e_{i,d} \in Path(v_l)} f_{i,d}(x_{a_j})$$

kde $Leaves$ označuje množinu všech listů a $Path(v_j)$ je množina všech hran na cestě z kořene v_1 do uzlu v_j.

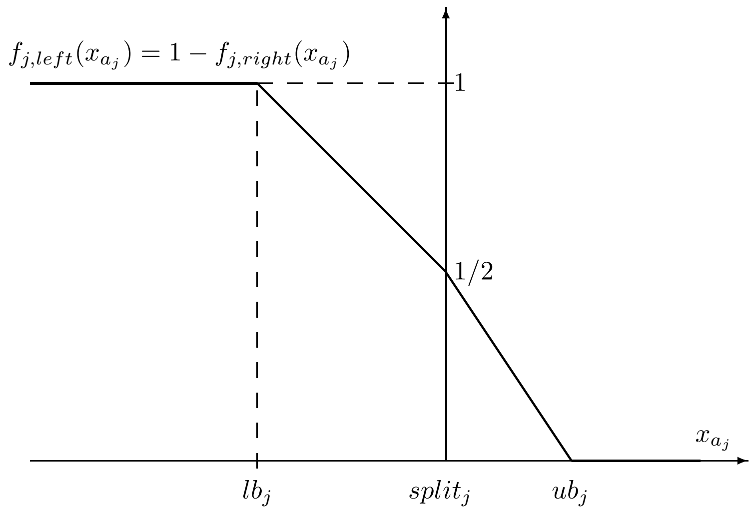
Optimalizace změkčení daného stromu potom znamená hledání tvarů změkčujících křivek. Přitom oproti klasifikaci bez změkčení se predikce může změnit v bodech, kde je hodnota změkčující křivky

$$f_{j,left}(x_{a_j}) \neq 1 \quad \text{pro } x_{a_j} \leq c_j$$

respektive

$$f_{j,right}(x_{a_j}) \neq 1 \quad \text{pro } x_{a_j} > c_j$$

Je výhodné, aby změkčující křivky byly parametrizovány tak, že jejich tvar na jedné straně od rozdělující hodnoty c_j lze změnou parametrů měnit aniž by se měnil tvar na druhé straně od rozdělující hodnoty. Díky tomu bude změnou jednoho parametru změkčení změněna predikce u jednodušeji strukturované množiny vzorů — všechny budou v jednom poloprostoru určeném původní rozhodovací podmínkou. Jinak řečeno,



Obrázek 1: Změkčující křivka

umožňuje to postihnout změkčením oblast na jedné straně od dělící nadroviny a přitom zachovat klasifikaci na opačné straně, pokud např. je již vyhovující.

Z tohoto důvodu jsme pro další experimenty zvolili změkčující křivky jejichž tvar je na Obrázku 1. Jedná se o spojité křivky po částech lineární, které se lámou v bodech lb_j (lower bound), c_j a ub_j (upper bound). Hodnota c_j je dána z výchozího stromu, hodnoty lb_j a ub_j jsou parametry křivky, přičemž platí

$$\forall j \quad lb_j \leq c_j \leq ub_j \tag{2}$$

Změkčování hran ve stromu $\mathcal{T}$ je potom úlohou optimalizace parametrů lb_j, ub_j, kde j prochází indexy všech vnitřních uzlů stromu $\mathcal{T}$ tak, aby strom se změkčujícími křivkami určenými těmito parametry poskytoval nejlepší predikci na trénovací množině.

3. Optimalizace změkčení

Byly provedeny experimenty zjišťující efekt změkčení hran. V těchto experimentech jsme použili optimalizační metody pro hledání minima funkce, která byla parametrizována stromem $\mathcal{T}$ a trénovací množinou a jejímž argumentem byl vektor $2m$ reálných čísel, kde m označuje počet vnitřních uzlů stromu $\mathcal{T}$. Tento vektor obsahoval hodnoty lb_j, ub_j pro všechny vnitřní uzly. Funkční hodnotou byl počet chyb při klasifikaci trénovací množiny stromem $\mathcal{T}$ se změkčením určeným parametry obsaženými v argumentu.

Protože použité implementace optimalizačních metod neumožňovaly omezení definičního oboru, přitom však přípustné změkčení určují jen takové argumenty, při kterých je splněna nerovnost (2), byla optimalizovaná funkce definována tak, že pokud argument určoval přípustné změkčení stromu $\mathcal{T}$, funkční hodnota byla rovna počtu chybných klasifikací stromem $\mathcal{T}$ s tímto změkčením na trénovací množině, pokud argument nebyl parametrizací přípustného změkčení, potom funkční hodnota byla o 1 větší, než velikost trénovací množiny. Jako iniciální hodnota argumentu pro optimalizaci byl použit vektor, který obsahoval parametry

$$\forall j \quad lb_j = c_j = ub_j \tag{3}$$

což definuje „nulové změkčení", tedy takové, že strom s tímto změkčením dává stejné predikce, jako strom bez změkčení. Zároveň je tato iniciální hodnota průnikem hraničních nadrovin vymezujících přípustné změkčení (2).

Protože optimalizovaná funkce závisí na všech vzorech z trénovací množiny, vykazuje velkou „členitost” a má značné množství lokálních minim, což je třeba vzít v úvahu při výběru optimalizační metody. Testován byl simplexový algoritmus pro minimalizaci [4], a simulované žíhání [1]. Simplexový algoritmus nemá prostředky k vyváznutí z lokálních minim a změkčení dosažená tímto algoritmem klasifikovala data podstatně hůře, než změkčení získaná metodou simulovaného žíhání, jež je díky randomizaci proti lokálním minimům mnohem odolnější.

Použité optimalizační metody pro správnou funkčnost vyžadují, aby ve vstupním prostoru měly všechny dimenze stejnou škálu[2], tedy aby jednotkový krok v libovolném směru měl vždy přibližně stejný význam. V našem případě jednotlivé dimenze parametru určují hranice oblastí změkčení u jednotlivých podmínek v rozhodovacím stromu, neboli u hranic hyperkvádrů. Rozhodovací strom určuje hierarchii hyperkvádrů — v každém vnitřním uzlu stromu je jeden hyperkvádr rozdělen na dva hyperkvádry nižší úrovně. Nabízí se tedy možnost odvozovat škálu prostoru parametrů od velikosti hyperkvádrů vyšší úrovně v příslušných směrech následovně: Nejprve celý prostor ve všech směrech omezíme nejzazšími trénovacími vzory, tak získáme základní hyperkvádr. Když v uzlu v_j podmínka (1) rozděluje hyperkvádr vyšší úrovně, který je v proměnné x_{a_j} omezen hodnotami b_1, b_2, kde $b_1 < c_j < b_2$, potom za jednotkový krok v parametru lb_j resp. ub_j považujeme $c_j - b_1$ resp. $b_2 - c_j$.

Je však pravda, že na hodnotu optimalizované funkce má vliv počet trénovacích vzorů zahrnutých do oblasti změkčení, který závisí kromě velikosti oblasti změkčení také na hustotě trénovacích vzorů v této oblasti. Proto by se měla škála odvozovat také od rozmístění vzorů. V dosavadních experimentech použití škály založené na mediánu případně průměru vzdálenosti bodů v odděleném hyperkvádru od dělící nadroviny vedlo k výsledkům podobným jako použití škály založené na velikosti hyperkvádrů. Tato oblast je předmětem dalšího zkoumání.

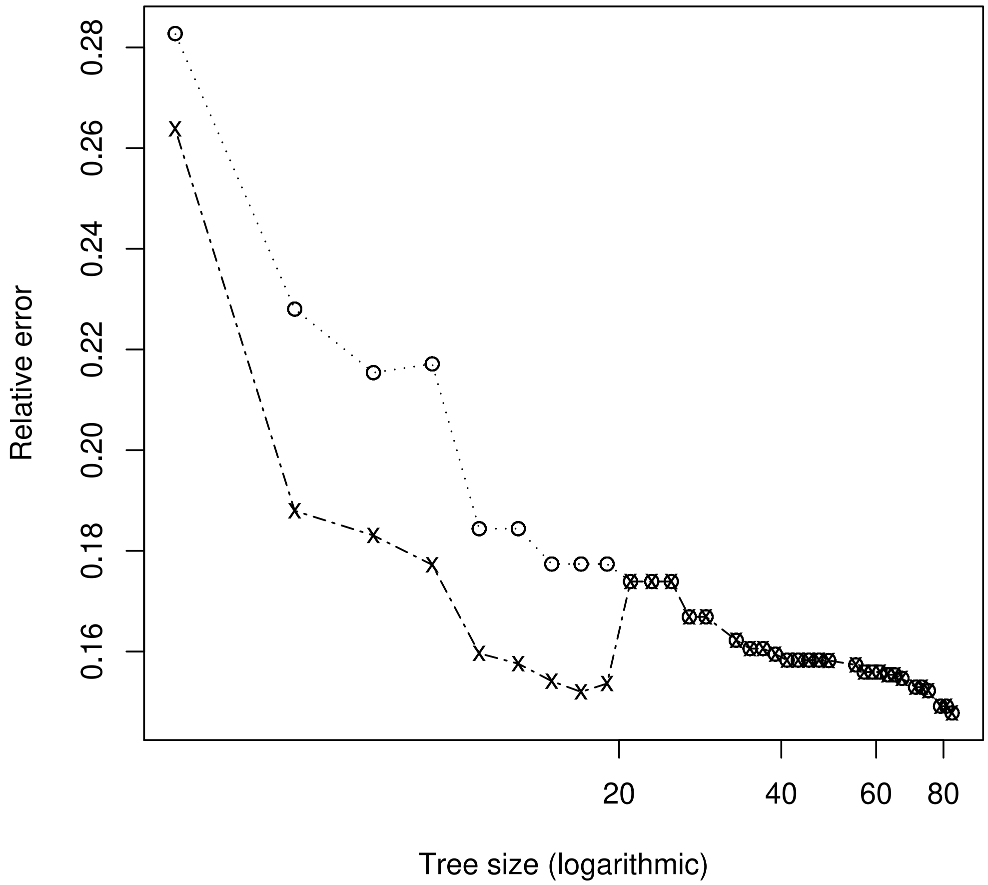
4. Výsledky experimentů

Experiment byl proveden na datech „Magic Telescope”[2], což jsou data s 10 reálnými atributy klasifikovaná do 2 tříd. Trénovací množina obsahovala 12679 vzorů, data byla standardizována průměrem a směrodatnou odchylkou. Trénovací množina byla rozdělena na dvě části stejné velikosti, jedna část byla použita pro vytvoření stromu a druhá část jako validační množina pro prořezávání. Strom byl vytvořen metodou CART a potom postupně prořezáván postupem používaným v metodě CART (viz [3]). Po každém prořezání bylo hledáno změkčení s použitím simulovaného žíhání vůči celé trénovací množině. Protože simulované žíhání je nedeterministická metoda, změkčení bylo hledáno 10-krát a potom vybráno nejlepší.

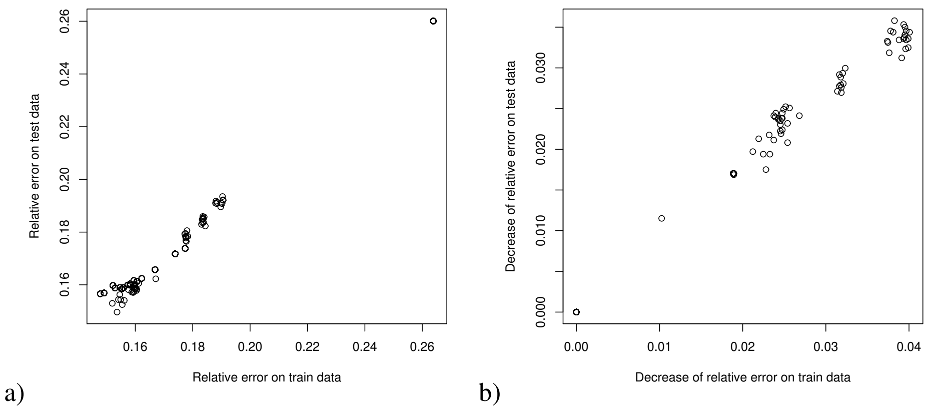
Výsledky ukazuje Obrázek 2, na horizontální ose je velikost stromu v logaritmickém měřítku, kroužky vyznačené body zobrazují relativní počet chyb na trénovací množině při použití stromu bez změkčení, křížky označují chybu stromu se změkčením. Body jsou pro lepší názornost spojeny čarami. Z obrázku je patrné, že stromy se změkčenými hranami o méně než 20 uzlech mohou dosahovat lepší kvality predikce než stromy bez změkčení velikosti 60, což zejména v případě, že chceme klasifikátor interpretovat do formy srozumitelné člověku, je úspora značná. U stromů o více než 20 uzlech nebylo nalezeno změkčení, které by vedlo ke zlepšení klasifikace. Je to způsobeno tím, že dimenze optimalizované funkce je potom také větší než 20 a vzhledem k tomu, že iniciální hodnota optimalizace (3) je ve vrcholu kuželu přípustných hodnot, je pravděpodobnost, že jeden krok simulovaného žíhání postoupí do oblasti přípustného řešení příliš malá (menší než 2^{-20}).

V experimentu provedený výběr z několika změkčení toho nejlepšího vzhledem k trénovací množině se ukazuje jako legitimní vzhledem k silné korelaci zlepšení na trénovací množině a na testovací množině, což ukazuje Obrázek 3 (testovací množina obsahovala 6341 vzor).

[2]Metodám sice může být předána funkce, která tento požadavek nesplňuje a škála může být předána jako zvláštní parametr. To je ovšem pouze formální rozdíl, jde totiž o to, že škála musí být v obou případech známa.



Obrázek 2: Relativní chyba klasifikace původním stromem a stromem se změkčením při postupném prořezávání



Obrázek 3: a) Souvislost relativní chyby na trénovacích a testovacích datech u stromů se změkčením b) Souvislost poklesu relativní chyby díky změkčení na trénovacích a testovacích datach

5. Závěr

Ukázali jsme v tomto článku některé vlastnosti změkčování hran v rozhodovacích stromech, pokud je chápáno jako úloha minimalizace počtu chyb na trénovací množině v závislosti na parametrech změkčení pro předem daný strom. Prozkoumání vlivu parametrů změkčení v jednotlivých hyperkvádrech prostoru rozděleného zadaným stromem nás vedlo k parametrizaci změkčení pomocí změkčujících křivek, jejichž části lze obměňovat různými parametry bez vzájemné závislosti. Uvedli jsme, že chaotičnost optimalizované funkce vyžaduje použití optimalizační metody odolné proti uváznutí v lokálním minimu. Pro experimenty byla zvolena metoda simulovaného žíhání. Určení škály prostoru parametrů změkčení aby se vyznačoval rovnoměrností v různých směrech je předmětem výzkumu, zmíněny byly škály založené na rozměrech hyperkvádrů a na rozložení trénovacích dat.

V experimentech jsme ukázali, že změkčování hran vede ke zlepšení klasifikace rozhodovacím stromem a že velikost zlepšení na trénovací množině je dobrým ukazatelem zlepšení na testovacích datech, takže je legitimní z několika různých změkčení vybrat nejlepší pouze s použitím trénovacích dat.

Literatura

[1] C.J.P. Belisle, “Convergence theorems for a class of simulated annealing algorithms on Rd.”, *J. Applied Probability*, vol. 29, pp. 885-895, 1992.

[2] R.K. Bock, A. Chilingarian, M. Gaug, F. Hakl, T. Hengstebeck, M. Jiřina, J. Klaschka, E. Kotrč, P. Savický, S. Towers, A. Vaicilius, “Methods for multidimensional event classification: a case study using images from a Cherenkov gamma-ray telescope.” *Nucl. Instr. Meth.*, A 516, pp. 511-528, 2004

[3] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, *Classification and Regression Trees*, Belmont CA: Wadsworth, 1993

[4] J.A. Nelder and R. Mead, “A simplex algorithm for function minimization.”, *Computer Journal* vol. 7, pp. 308–313, 1965.

[5] J.R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, San Mateo — California, 1993

Behavior Protocols: Efficient Checking For Composition Errors

Post-Graduate Student:
MGR. JAN KOFROŇ

Faculty of Mathematics and Physics
Charles University in Prague
Malostranské náměstí 25
118 00 Prague 1
Czech Republic

kofron@nenya.ms.mff.cuni.cz

Supervisor:
PROF. ING. FRANTIŠEK PLÁŠIL, DRSC.

Faculty of Mathematics and Physics
Charles University in Prague
Malostranské náměstí 25
118 00 Prague 1
Czech Republic

plasil@nenya.ms.mff.cuni.cz

Field of Study:
Software Systems
Classification: I2

Abstract

Checking components behavior compatibility when using behavior protocols as a specification platform brings, besides others, the state explosion problem. This problem lowers the complexity of protocols that can be checked on rather low level. The problem lies not only in high time requirements, but also in keeping information about visited states in the memory. To cope with this problem, we present bit-based state representation enabling for keeping a substantially higher number of states identifiers in the memory. Thus, it is possible to check protocols describing almost all real-life components.

1. Introduction

High reliability is a property is required more and more often from a number of today's computational systems. This requirement originates in the use of modern technologies in the areas of surgery, avionics, but also banking industry. To assure absence of errors, the system has to undergo a process of formal verification. One way to prove that a system satisfies a set of properties (usually absence of certain errors) is model checking. To perform model checking (i.e. checking for validity of a system property), a formal description of both the system and the property is needed. The system in conjunction with the tested property usually yields a huge state space that has to be exhaustively traversed. This problem is known as the state explosion problem [3].

As well as in model checking [3], behavior protocols also have to face the state explosion problem. The state explosion, i.e. enormous (exponential) growth of the system state space, causes high time and memory requirements of these methods and hinders their everyday usage as a method for finding bugs in software.

Besides the time requirements, problems lie also in keeping information about what parts of the state space has been already visited. One cannot profit from the secondary memory usage, because the overall time needed to access this type of memory for storing states' information exceeds all acceptable limits. The problem of state space explosion is inherent to model checking and today's computational systems are not powerful enough to successfully cope with it.

The goal of our work is to develop methods that would allow checking of behavior protocols [1] of all components; recently, components featuring complex behavior protocols couldn't be checked due to high time and memory requirements.

1.1. State Space Explosion

In order to make the traversal of large state spaces possible, it is necessary not to generate the whole state space before the computation. Therefore, state space generation strategy called on-the-fly is to be used [3]. Using this strategy, the states and transitions interconnecting these states are not pregenerated at the beginning of the checking process, but each time (i.e. in a state) a transition (or another state) needs to be explored, it is generated. The structure of already visited part of the state space can be forgotten saving thus the memory. On the other hand, it is desirable to recognize these already visited parts, in order no to visit them more than once, so a kind of information about state visits has to be kept along the checking process.

Sometimes, the state space to be traversed may be so huge that even holding this kind of information within the memory is not feasible on today's systems. There are basically three options to solve this: (i) To "forget" information about some states visited when the memory becomes full, (ii) use approximate information about states, and (iii) use a symbolic state space representation. Although very fast and memory saving, the second option may cause omitting traversal of some parts of the state space lowering thus the checking result's reliability [4]. The third option requires further research of applicability of these methods (e.g. OBDDs) in the scope of behavior protocols and currently we are not sure of their contributions. Thus, we decided to focus on the first option.

1.2. Behavior Protocols

SOFA behavior protocols [1] are a mechanism for component behavior specification. They were originally developed for description of SOFA components [5], but can be extended for behavior description in a wide variety of component models; they describe component behavior on the level of method calls and method calls' responses.

In SOFA, component behavior is described by the component frame - a black-box view that defines provided and required interfaces of the component. The architecture of a composed component describes its structure at the first level of nesting - the connections between subcomponents and the bindings of the component interfaces to the interfaces of the subcomponents.

The component behavior is defined by a set of possible traces, i.e. sequences of events (method calls and returns from methods) the component is allowed to perform. Since this set might be quite huge or even infinite, the explicit enumeration of its members becomes impossible. Thus, the behavior protocols, i.e. the expressions generating all traces allowed to be executed, are used. Besides standard regular-expression operators (e.g. ';' for sequencing, '+' for alternative and '*' for repetition), there are also special operators like the and-parallel operator. The and-parallel operator generates an arbitrary interleaving of all events within its operands (e.g. `(a ; b) | (c ; d)` protocol generates those six traces starting with 'a' or 'c' where 'a' precedes 'b' and 'c' precedes 'd').

As an example consider the following protocol:

```
(open ; (read + write)* ; close) | status*
```

This behavior protocol generates the set of those traces starting with the `open` method and ending with the `close` method with an arbitrary sequence of repeating the `read` and `write` methods between them. The traces may be interleaved with an arbitrary number of the `status` method calls.

Behavior protocols allows for testing protocol compliance and absence of composition errors. Since the compliance (i.e. conformance of a component frame protocol with its architecture protocol) is realized as a composition test of the architecture protocol and the inverted frame protocol [6], only the checking for composition errors will be described in more detail.

The composition test allows for detection of three error types: *bad activity*, *no activity*, and *infinite activity*. Bad activity denotes the situation when a call is emitted on a requires interface, but the component bound via its (provides) interface to this required interface is currently (according to its frame protocol) not able to accept that call. No activity basically denotes deadlock, i.e. the situation when (i) none of the components is able to perform any action and (ii) at least one component is not in one of its final states. Situation where the components input a loop without a possibility to reach a final state (i.e. there is no trace from any loop state to a final state in the state space) is called infinite activity.

Size of the state space generated by protocols describing behavior of a composed component tends to grow exponentially when using a type of parallel composition, i.e. the and-parallel and consent operators.

Evaluating both compliance and composition relations requires exhaustive state space traversal. Thus, coping with the state space explosion problem is inevitable.

2. Behavior Protocol Checker

The behavior protocol checker is a tool performing compliance and composition tests of communicating components' behavior described by behavior protocols. It uses *Parse Tree Automata* (PTA) [2] for the state space generation; PTAs are subject to various optimizations decreasing the size of the state space they generate. These optimizations involve *Multinodes*, *Forward Cutting*, and *Explicit Subtree Automata* [2].

The *multinodes* optimization is applied on the parse tree during its construction (when parsing the input protocol). It replaces the repeating occurrences of the same protocol operator by a single occurrence of multi-version of the same operator, i.e. the sequence of n-1 occurrences of an operator is replaced by one n-ary version of the same operator.

The *forward cutting* optimization is applied after the parse tree construction and it inheres in iterative deleting those leaves of the parse tree that are discarded by the restriction operator present in a higher level of the parse tree.

The last optimization applied on the parse tree is the *explicit subtrees* optimization. This optimization converts some subtrees of the parse tree into an explicit automaton. This means that for this parse subtree the states are not generated on-the-fly, but they are precomputed before the state space is traversed. There are two criteria for selecting subtrees to be converted to explicit automata: (i) The subtree has to have a reasonable size to fit in the memory and (ii) its states have to be used more than once (e.g. in subtrees of an *and-parallel* operator) during the composition test. As it is necessary to traverse the whole state space generated by the subtree to construct the explicit automaton, there will be no benefit from converting this part to an explicit automaton if its states are used only once. On the other side, there must be enough free memory left for future storing information about visited states.

It is almost obvious that none of the optimization described above affects the language (the set of traces) generated by the PTA. Although the multinode and explicit tree optimizations, unlike forward cutting, don't reduce the state space, they increase the checking speed in a significant way in the sense of higher transition generation speed.

Sometimes, however, even after applying these optimizations, the size of the resulting state space may still exceed the limits of the system the test is performed on. Therefore, suitable methods for large state space traversal have to be used.

2.1. Fastening the State Space Traversal

The use of the *Depth First Search with Replacement* traversal technique (DFSR) [2] (similar to DFS, but it forgets information about some state visits when the available memory becomes full) may result in traversing a significant part of the state space more than once. This, obviously, negatively affects the checking time requirements. To successfully cope with this problem, it is necessary to maximize the number of state identifiers storable in the memory; thus, compact state identifiers are needed.

As mentioned in the previous section, PTAs are used for the state space generation. PTA representing a behavior protocol is a tree isomorphic to the protocol parse tree, extended as follows: (i) in each leaf there is a primitive automaton representing an event of the protocol (i.e. it has two states - an initial and a final one) and (ii) each inner node combines its children PTAs using a protocol operator. We can then exploit the fact that a state within a state space is uniquely determined by the states of all the leaf automata of PTA. Since each leaf automaton has only two states, it is natural to represent its state by a single bit. The leaf automata state identifiers are then combined in a given order (regarding the position in the parse tree) and enriched by necessary information from some inner nodes. The resulting bit-field represents a state of the whole PTA. The information added to a state identifier from an inner node includes, for instance, information about which "branch" of an alternative node (representing the alternative operator) carries valid information about this state (exactly one branch of an alternative node may be valid in a particular state).

As an example consider PTA corresponding to the protocol from Section 1.2 on Figure 1. The state identifier representing the initial state of the behavior protocol corresponding to this PTA has the form `001100000`. The leading two zeros represent initial states of the primitive automata for `open`↑ and `open$`, two following ones express that none of the branches of the alternative ('+') operator has been selected so far, the following zero (redundant) represents the state of the alternative subtree (to simplify and fasten the identifier manipulation) and last four zeros represent initial states of primitive automata for `close`↑, `close$`, `status`↑ and `status$`. Having the PTA, the information about inner nodes type (except for the alternative operator) needn't to be stored inside an identifier. Similarly, the identifier of a final state of this protocol has the form `111101111` (neither the `read` nor `write`, unlike the `status` operation, were executed in the trace corresponding to this final state, thus, the automaton for alternative subtree is still in its initial state - `110`). The size of state identifiers is 9 bits in this case. To compare it with the size of previous state identifiers representation, the former identifiers denoting the initial and accepting states were `ci0s0i0s0` and `ci3i2s3i2s3`, respectively (9 and 11 BYTES), being thus at least eight times larger. The letters inside of the original state identifiers denotes the inner node types reflecting their relation to the associated type of state; for instance 'c' stands for composite state expressing that all of corresponding node subtrees creates the state (e.g. the parallel operator '|'), while index state ('i') denotes the only subtree creating this state (e.g. alternative operator '+').

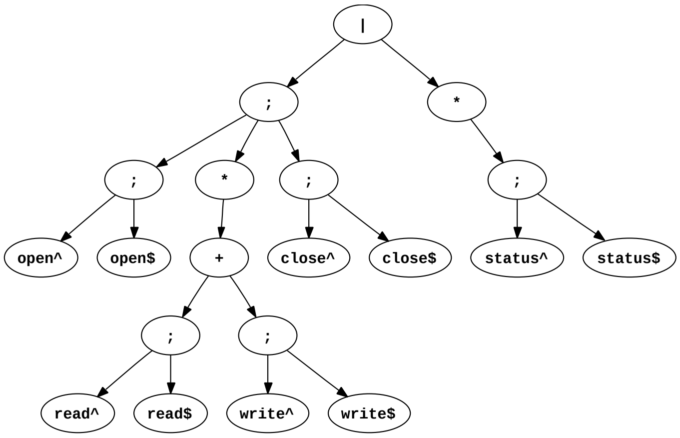


Figure 1: Protocol parse tree

In the case PTA is deterministic, the proposed state identifiers are all of the same size and this fact can be exploited in state identifier memory allocations. However, PTAs are non-deterministic in general, which slightly complicates the state identifiers' memory management (reallocations take place from time to time in a way trying to keep the time and memory requirements balanced). While traversing the state space, state identifiers (of entire PTA) are stored in memory (in a data structure called "state cache") and, as they represent a lossless state representation, they are used for state space generation (generation of transitions from the state an identifier represents). When the memory dedicated for the state cache use becomes full, a randomly chosen subset of states stored in the state cache is purged from the cache letting thus new states to be stored.

3. Evaluation

Although the original representation enabled representing states of the same complexity (i.e. states of the same input behavior protocol) in a context of slightly simpler (and smaller) parse tree (the leaves could represent more complex automata), the new representation is approximately eight times more efficient. This implies not only a larger number of state identifiers storable in the memory, but decreases also checking time requirements in two aspects: (i) the state identifiers comparison is much faster, and (ii) the state identifiers need not to be "forgotten" so often, therefore larger state spaces can be traversed without states forgetting (e.g. purging of the state cache), thus in a reasonable time. Hence, when using the new state identifiers, most of real-life protocols can be checked for their compliance with each other and absence of composition errors.

4. Related Work

The other methods for fighting the state explosion problem include *Partial Order Reduction* [3], using symbolic state representation, e.g. *Ordered Binary Decision Diagrams* (OBDDs) [3], parallel state space traversing, and *Bit-state Hashing* [4]. Although these techniques have been more or less successfully used in model checking in last years, applying the partial order reduction and using OBDDs for state space representation has showed to be very difficult (and probably not beneficial) in the case of behavior protocols. Bit-state hashing and parallel state space traversing are methods that we believe can be applied to the behavior protocol checking and are subject of future research.

5. Conclusion and Future Work

Although very beneficial, the on-the-fly state generation strategy and DFSR traversing in conjunction with bit-based state identifiers still don't allow for model checking of all real-life behavior protocols due to the enormously high size of states spaces (often in the order of 10^8). In these cases, when an exhaustive state space traversal is impossible with current methods, an approximate technique (e.g. bit-state hashing) will be implemented as it may still provide a piece of useful information. Furthermore, research regarding the state cache purging (now a random eighth of the state cache is purged) will be done, although now it seems to be very hard to develop an purging algorithm for general case, since no useable information about the structure of the state space to be traversed is at the moment of purging available.

The future work will therefore focus on further improvements of state representation and increasing of the checking speed using hashing and parallel state space traversing, as well as on generalization of the methods for use in other model checkers (e.g. Spin [4]).

References

[1] Plášil, F., Višnovský, S.: Behavior Protocols for Software Components, IEEE Transactions on Software Engineering, vol. 28, no. 11, Nov 2002

[2] Mach, M., Plášil, F., Kofroň, J.: Behavior Protocol Verification: Fighting State Explosion, published in International Journal of Computer and Information Science, Vol.6, Number 1, ACIS, ISSN 1525-9293, pp. 22-30, Mar 2005

[3] Edmund M. Clarke, Jr., Orna Grumberg, Doron A. Peled: Model checking, The MIT Press, 1999

[4] Gerard J. Holzmann: The Spin model checker: Primer and Reference Manual, Addison-Wesley, 2003

[5] SOFA Component Model, http://sofa.objectweb.org

[6] Adámek, J., Plášil, F.: Component Composition Errors and Update Atomicity: Static Analysis, accepted for publication in the Journal of Software Maintenance and Evolution: Research and Practice, 2005

Execution Language for GLIF-formalized Clinical Guidelines

Post-Graduate Student:
MGR. PETR KOLESA
EuroMISE Centrum – Cardio
Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2
182 07 Prague 8
kolesa@euromise.cz

Supervisor:
DOC. ING. ARNOŠT VESELÝ, CSC.
EuroMISE Centrum – Cardio
Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2
182 07 Prague 8
vesely@euromise.cz

Field of Study:
Biomedical Informatics
Classification: 3918V

The work was partly supported by the project no. 1ET200300413 of the Academy of Sciences of the Czech Republic and by the Institutional Research Plan no. AV0Z10300504.

Abstract

This article is on obstacles we faced when developing an executable representation of guidelines formalized the Guideline Interchange Format (GLIF). The GLIF does not fully specify the representation of guidelines at the implementation level as it is focused mainly on the description of guideline's logical structure. Our effort was to develop an executable representation of guidelines formalized in GLIF and to implement a pilot engine, which will be able to process such guidelines. The engine has been designed as a component of the MUltimedia Distributed Record system version 2 (MUDR^2). When developing executable representation of guidelines we paid special attention to utilisation of existing technologies to achieve the highest reusability.

Main implementation areas, which are not fully covered by GLIF, are a data model and an execution language. Concerning the data model we have decided to use MUDR^2´s native data model for this moment and to keep watching the standardisation of a virtual medical record to implement it in execution engine in the near future. When developing the execution language, first of all we have specified necessities, which the execution language ought to meet. Then we have considered some of the most suitable candidates: Guideline Execution Language (GEL), GELLO, Java and Python. Finally we have chosen GELLO although it does not completely cover all required areas. The main GELLO's advantage is that it is a proposed HL7 standard. In this paper we show some of the most important disadvantages of GELLO as an executable language and how we have solved them.

1. Introduction

One of the main research projects running at EuroMISE Centre is the development of clinical decision support systems. The aim of this effort is to develop tools that will reduce routine activities performed by the physician as well as allow standardisation and improving of given care, which implies substantial financials savings. Many projects focused on formalization of the knowledge contained in clinical guidelines were recently running at EuroMISE Centre. It is known that only a small number of physicians uses clinical paper-based guidelines during diagnosing and treatment. This problem is caused by the existence of a large amount of relevant guidelines, which are frequently updated in addition . It is very time-consuming to monitor all relevant guidelines. We have experience formalizing clinical guidelines using the Guideline Interchange Format (GLIF) [1]. Guidelines formalized in such a way have been used mainly for education of medical students, but a tool for automatic execution of formalized guidelines [2], [3] did not exist.

Another important project recently realized at EuroMISE Centre was the development of a new system for structured electronic health record called MUltimedia Distributed Record version 2 (MUDR^2) [4]. One of the MUDR^2's most valued features is that it allows to change the data model dynamically and on-fly [5]. In addition, it supports versioning of the data model, so the access to the data is possible not only by the current date, but also at an arbitrary point in the past. This feature is important for checking whether a treatment conforms to the guidelines.

2. Materials and Methods

We have decided to join outputs from both these projects to create a guideline based decision support system. We have created an execution engine that is able to process formalized guidelines and added it to MUDR^2 system. MUDR^2 already contained some decision support system tools. These tools could be extended by plugins, but the knowledge gained from guidelines was required to be hardwired in the program code of plugins. This approach was not convenient, since it was difficult and expensive to change a plug-in, when a certain guideline had been updated. The utilisation of GLIF-formalized guidelines solves this problem. As the GLIF does not specify the guideline's implementation level, we have developed our own executable representation of the GLIF model.

When developing an executable representation of guidelines we paid special attention to utilisation of existing technologies and tools wherever possible. We have already converted some guidelines to the GLIF model in Protege 2000 [6]. These guidelines are transferred to an executable representation in XML. It is also possible to export a formalized guideline from Protege to the Resource Description Framework [7] (RDF) format. This export depicts static structure very well, but there are few possibilities to represent a dynamic behaviour like data manipulation. Therefore we have developed our own representation, which is based on XML and has all features missing in the Protege RDF export.

From the point of implementation, two areas of executable representation are important: the data that are processed (data model) and the program that works with these data (execution language). In the next sections we will describe them in greater details.

3. Results

3.1. The Data Model

The data model must define following three topics: which data it provides, a data naming convention and a structure of these data. An obvious choice of the data model for clinical guidelines is a virtual medical record (VMR) [8]. As the standard of the VMR is still being developed, we have decided for now to use the native data model of MUDR^2 and keep watching the standardisation process to be able to implement the VMR in the future.

3.1.1 Which Data It Provides: As the MUDR^2 allows to represent arbitrary data in its data model, it does not fully specify which kind of data it contains. For the purposes of guidelines we can use the minimal data model (MDM), which is specified for certain medical specialties. The MDM for a medical specialisation is an agreement made by specialists on the minimal data set, which is necessary to know about a patient in the medical specialisation, for instance Minimal Data Model for Cardiology [9].

3.1.2 The Naming Convention: The naming convention specifies the name of certain data. The naming convention of the MUDR^2 data model has its origin in the logical representation of this model [6]. Data are represented as an oriented tree, where each node has a name. A certain piece of data is addressed by its full name, which is build of nodes' names on the path from the root to the given node.

3.1.3 The Structure of Provided Data: The data structure specifies which information is carried by a data item (a node in MUDR^2). A data item in MUDR^2 contains a value and a unit of measurement.

Further, it contains some administrative information about when the item was created, by whom it was created, how long it will be valid, etc.

3.2. The Execution Language

The execution language is another important issue of the executable representation of guidelines. All expressions contained in a formalized guideline are written using this language. An execution language consists of four sublanguages with different roles.

- The arithmetic expression sublanguage is a language for algebraic formulae and for the control of the program flow.
- The query sublanguage is probably the most important part of the execution language. Its role is to retrieve the requested data from an underlying data model. It must meet two conflicting requirements: sufficient language's power and performance. SQL, for instance, is a good example of a suitable query sublanguage.
- The data manipulation sublanguage contains constructs for non-trivial data transformations like manipulation with strings, converting data from one format to another (e.g. sorting of a sequence by a given attribute), etc.
- The date related functions. Although it is not a real sublanguage, we mention it separately, because it is very important for medical guidelines. It must be possible to write expressions like three month ago or to count a difference between two dates, and to represent this difference in arbitrary time units (days, months, etc.). These operations must conform to the common sense, e.g. a month is from 28 to 31 days long depending on the context.

When choosing the execution language, we considered four languages: GEL, GELLO, Python and Java. They are shortly described below and theirs pros and cons are mentioned.

- GEL is an execution language defined in the GLIF specification version 3.5 [1]. GEL is not object-oriented and it does not contain a query sublanguage. The data manipulation language is in some aspects very limited (e.g., it is impossible to order sequences according to other attributes then time.). At this state of development, GEL is not suitable to be an execution language.
- GELLO [10] is an object-oriented successor of GEL. It contains a query language, which is semantically very similar to SQL. The data manipulation language is more powerful than in GEL, but it still lacks some important features, e.g. working with strings. Furthermore, GELLO does not contain any date related functions, which have to be provided by the underlying layers. A great advantage of GELLO is that it is proposed to be an HL7 standard.
- Java is an all-purpose modern programming language. The advantage is that Java is a widespread language and it contains all sublanguages mentioned above. It is easily extendable through packages. In addition, many compilers and interpreters are available. Moreover, there is an embeddable interpreter called BeanShell [11]. Beside this, Java's standard packages java.util.Date and java.util.Calendar meet all requirements on data and time processing. A disadvantage is that Java is too general and it is more complicated to use it in guidelines than any specialised language.
- Python has an advantage over Java – though it is a procedural language, it has many functional features. That is useful in the knowledge representation. But compared to Java, there is a major disadvantage: datetime, a build-in module for working with time and date, does not fully comply with the above stated requirements (the months in datetime module have always 30 days, etc.) Other pros and cons are the same as in Java.

Although GELLO does not meet all requirements, we have finally chosen it as the execution language, mainly because it is a proposed HL7 standard. Then we solved how to cope with the features it lacks. Specification of GELLO left all things that GELLO does not solve to the technologies or languages in the sublayers. In this case a programmer formalizing clinical guidelines would have to master another formal language in addition to GELLO and GLIF. We have decided not to solve missing features by another language, which would appear in GELLO´s expressions, and we have introduced all necessary extensions directly into GELLO. The extensions are described in the rest of this section.

First, GELLO does not contain constructs that make the code maintenance significantly easier and that are usually present in modern programming languages. These missing constructs are for instance the typedef and the function (or similar) construct. We have added both of them to the execution language.

Second, neither GLIF nor GELLO specify the scope of variables. There exist three logical scopes in formalized guidelines: an entire guideline, a subgraph and a step. By adding the function construct there is another scope: a body of the function. There is no need to have a variable with scope of the entire guideline, since each guideline consists of one or more subgraphs and GLIF specifies that subgraphs interact only through the in, out and inout variables. It is useful to distinguish among the remaining three scopes. To do so we have added three keywords that are to be used when defining a new variable with either the function scope, the step scope, or the graph scope.

Finally, we faced the problem of missing date functions. GELLO does not define any date or time functions. This functionality it is left to the underlying facilities. We have added all the necessary time constructs that were defined in GEL.

3.3. Encoded Guidelines

To verify that the described representation is suitable for clinical guidelines formalized in the GLIF model, we have converted six guidelines to the executable representation: European Guidelines on Cardiovascular Disease Prevention in Clinical Practice [12], Guidelines for Diagnosis and Treatment of Ventricular Fibrillation [13], 1999 WHO/ISH Hypertension Guidelines [14], 2003 ESH/ESC Guidelines for the Management of Arterial Hypertension [15], Guidelines for Diagnosis and Treatment of Unstable Angina Pectoris 2002 [16], Guidelines for Lung Arterial Hypertension Diagnosis and Treatment [17].

4. Discussion

There are some projects concerning the issues of computer-interpretable guidelines. The best known are Asbru [18] language and SAGE project [19].

Asbru has been developed at the University of Vienna as a part of the Asgaard project. The Asbru represents clinical guidelines as time-oriented skeletal plans. This is a very different approach compared to the GLIF's flowchart-based point of view. We believe that the flowchart-oriented approach is more natural to physicians than the Asbru's one. Thus we assume that flowchart-oriented decision support systems will be more acceptable for them.

The SAGE project is built up on GLIF's experience, but it uses the opposite approach to formalizing of guidelines (GLIF's top-down approach versus the from-the-bottom approach in SAGE). Unlike GLIF, SAGE specifies the implementation level of guidelines, but it gives up the aspiration of shareable guideline representation. As SAGE is still in the testing stage, it remains debatable which approach is more advantageous. Thus we have decided to utilise the guidelines that had been formalized in GLIF and we have converted them into the described execution language.

5. Conclusion

When developing an executable representation of guidelines formalized in the GLIF model, we realised that GLIF version 3.5 meets the needs of formalized guidelines very well.

During the conversion of formalized guidelines into the executable representation we have found out that the execution language GEL (a part of the GLIF 3.5 specification) lacks important features. Further, we have found out that GELLO, the successor of GEL, made considerable progress in bridging these gaps. However, GELLO still does not address some important features like date-related functions. Also the constructs that simplify maintenance of the code are still missing. We have added these features as an extension to the GELLO language. Further, we have developed a component that can process an executable representation of a guideline. This component is a part of the $MUDR^2$ system, which allows guidelines to use the data from structured health records stored in $MUDR^2$. Finally, we have converted six guidelines to the executable representation and verified that this representation is suitable for clinical guidelines formalized in the GLIF model.

References

[1] Mor P, Aziz B, Samson T, Dongwen W, Omolola O, Quing Z. "Guideline Interchange Format 3.5 Technical Spec." *http://smi-web.stanford.edu/projects/intermed-web/guidelines/GLIF_TECH_SPEC.pdf*, InterMed Collaboratory, 2002.

[2] Ram P, Berg D, Tu S, Mansfield G, Ye Q, Abarbanel R, Beard N. "Executing Clinical Practice Guidelines Using the SAGE Execution Engine." *In: MEDINFO 2004 Proceedings*, Amsterdam 2004, pp. 251–255.

[3] Wang D, Shortliffe EH. "GLEE – a model-driven execution system for computer-based implementation of clinical practice guidelines." *In: Proceedings AMIA Annual Fall Symposium*, 2002, pp. 855–859.

[4] Hanzlíček P, Špidlen J, Nagy M. "Universal Electronic Health Record MUDR. *In: Duplaga M, et al. eds. Transformation of Healthcare with Information Technologies*, Amsterdam: IOS Press, 2004, pp. 190–201."

[5] Špidlen J, Hanzlíček P, Říha A, Zvárová J. "Flexible Information Storage in MUDR2 EHR." *In: Zvárová J et al. eds. International Joint Meeting EuroMISE 2004 Proceedings*. Prague: EuroMISE Ltd., 2004, p. 58.

[6] Stanford Medical Informatics. "Protege 2000." *http://protege.stanford.edu/.*

[7] World Wide Web Consortium. "Resource Description Framework." *http://www.w3.org/RDF/*

[8] Johnson PD, Tu SW, Musen MA, Purves IN. "A Virtual Medical Records for Guideline-based Decision Support." *In: Proceedings AMIA Annual Fall Symposium*, 2001, pp. 294–298.

[9] Mareš R, Tomečková M, Peleška J, Hanzlíček P, Zvárová J. "User Interfaces of Patient's Database Systems – a Demonstration of an Application Designed for Data Collecting in Scope of Minimal Data Model for a Cardiologic Patient." *In: Cor et Vasa Brno*, 2002:44, No. 4, p 76. (in Czech).

[10] Sordo M, Ogunyemi O, Boxwala AA, Greenes RA. "GELLO: An Object-oriented Query and Expression Language for Clinical Decision Support." *In: Proceedings AMIA Annual Fall Symposium*, 2003, pp. 1012–1015.

[11] Been Shell. "Lightweight Scripting for Java." *http://www.beanshell.org/.*

[12] Backer GD, et al. "European Guidelines on Cardiovascular Disease Prevention in Clinical Practice." *In: European Journal of Cardiovascular Prevention and Rehabilitation*, 2003: 10: S1–S78 : 2003.

[13] Čihák R, Heinc P. "Guidelines for Diagnosis and Treatment of Ventricular Fibrillation." *http://www.kardio.cz/index.php?&desktop=clanky&action=view&id=83*. (in Czech: Doporučení pro léčbu pacientů s fibrilací síní).

[14] 1999 World Health Organisation – International Society of Hypertension "Guidelines for the Management of Hypertension." *In: Journal of Hypertension*, 1999, pp. 151–183.

[15] 2003 European Society of Hypertension – European Society of Cardiology "Guidelines for the Management of Arterial Hypertension." *In: Journal of Hypertension*, Vol 21, 2003, No. 6, pp. 1011–1053.

[16] Aschermann M. "Guidelines for Diagnosis and Treatement of Unstable Angina Pectoris." *http://www.kardio.cz/index.php?&desktop=clanky&action=view&id=86*. (in Czech: Doporučení k diagnostice a léčbě nestabilní anginy pectoris)

[17] Jansa P, Aschermann M, Riedel M, Pafko P, Susa Z. "Guidelines for Diagnosis and Treatement of Lung Arterial Hypertension." *http://www.kardio.cz/resources/upload/data/2_030_guidelines.pdf*. (in Czech: Doporučení pro diagnostiku a léčbu plicní arteriální hypertenze v ČR).

[18] Shahar Y, Miksch S, Johnson P. "An Intention-based Language for Representing Clinical Guidelines." *In: Proceedings AMIA Annual Fall Symposium*, 1996, pp. 592–596.

[19] Tu SW, Musen MA, Shankar R, Campbell J, Hrabak K, McClay J, Huff SM, McClure R, Parker C, Rocha R, Abarbanel R, Beard N, Glasgow J, Mansfield G, Ram P, Ye Q, Mays E, Weida T, Chute CG, McDonald K, Molu D, Nyman MA, Scheitel S, Solbrig H, Zill DA, Goldstein MK. "Modeling Guidelines for Integration into Clinical Workflow." *In: MEDINFO 2004 Proceedings*, Amsterdam 2004, pp. 174–178.

Time-convergence Model of a Deme in Multi-deme Parallel Genetic Algorithms: First Steps Towards Change

Post-Graduate Student:
ING. ZDENĚK KONFRŠT

Katedra kybernetiky
Fakulta elektrotechnická ČVUT
Technická 2
Praha 6

konfrst@zikova.cvut.cz

Supervisor:
ING. MARCEL JIŘINA, DRSC.

Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2
182 07 Prague 8

marcel.jirina@cs.cas.cz

Field of Study:
Umělá inteligence a biokybernetika
Classification: 3902V035

Abstract

Migration is an influential as well as important operator in multi-deme evolutionary algorithms. We investigate an time-convergence model of a deme. The model helps to understand it's significance and effect on the algorithm itself. As far as novelty is concern, we propose additions to migration equations to the known equations to investigate a broader scope of the algorithm behavior. To clarify and confirm it, evaluation tests of new features against real runs of the algorithms are done in the broad fashion.

1. Introduction

Multi-deme parallel genetic algorithms (PGAs) are parallel versions of sequential genetic algorithms in such a way that they do not run one population at the time, but many populations (demes) concurrently or in parallel. When they converge to a sub-solution, they exchange individuals with other deme and then the demes continue their runs.

The exchange is called migration and it is the basic term of this paper. Apart from migration, there are other terms like migrants, migration, migration rate, selection of migrants, migration frequency, migration intensity, replacing policies and takeover times. The terms are named intuitively, so no explanation is needed or complete definitions are in [2,3,4].

The article is organized as follows. Background section is the introduction to the research work on migration, migration policies and takeover times. Sections 3 is about replacement policies. In Section 4, the main focus is on the estimate of good individuals in a deme. Derivation of takeover times is in section 5. In Section 6, the improvements are combined in short manner. Section 6 concludes important findings of the paper.

2. Background

In the next section, we review the analysis [2, 4] of migration and takeover times for parallel genetic algorithm. Equations of the analysis were derived from a sequential case [1], that one we start with first.

Let P_t denote the proportion of good individuals in the population at time t and $Q_t = 1-P_t$ is the proportion of bad individuals. In case of tournament selection of size s, a bad individual survives if all participants in the tournament are bad ($Q_{t+1} = Q_t^s$). When we substitute $P = 1-Q$, the proportion of good individuals is $P_{t+1} = 1-(1-P_t)^s$. The extensions below presuppose that migration occurs every generation, it occurs after selection and it is clear that a close link from the previous derivations.

We should say that we have good G and bad B individuals in the population and they could be randomly R chosen and/or replaced. Let's define it as the third one, so the complete set of individuals is $U \in \{G, B, R\}$.

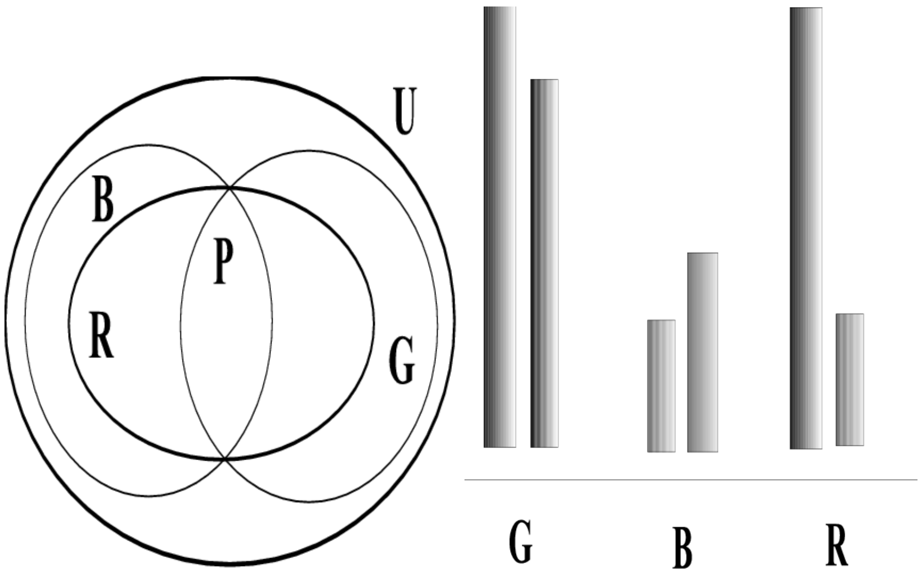


Figure 1: Diagram represents relations between the sets of individuals (left). Conditions between the sets are: $P = B \cap G \cap R = 0; R \in \{G, B\}; U \in \{G, B, R\}$. A fitness comparison between Good G, Bad B and Random R individuals (right). The size of a bar shows the size of individual fitness. One bar represents one individual. One can see that a bigger bar means higher fitness. Random set R has good and bad individuals.

3. Replacement policies

When *demes* exchange individuals one must decide how many individuals will migrate from the *sending deme* and how they replace individuals at the *receiving deme*. The analysis expects a migration every generation. Equations below give the proportion of good individuals P_{t+1} at the current generation $t+1$ based on the proportion of good individuals P_t from the previous generation t. At the sending deme, the size s of tournament selection defines how many individuals were chosen for migration. The migration rate is m. When good migrants replacing bad individuals, we define this as $(G \vdash B)$ and similarly for other cases.

3.1. Good migrants replace bad individuals: $(G \vdash B)$

When good migrants replace bad individuals, the proportion of good individuals increases by migration rate m. So just m was added to the above mentioned sequential case,

$$P_{t+1} = 1-(1-P_t)^s + m. \tag{1}$$

3.2. Good migrants replace random individuals: $(G \vdash R)$

When good migrants replace individuals randomly, we are interested how many bad individuals are replaced. Replacement of good ones by good ones does not change the situation. The probability of bad ones is equal to their proportion after selection $Q_{t+1} = Q_t^s = (1-P_t)^s$ and so the proportion of bad ones that is replaced by good migrants is $m.(1 - P_t)$. So we get

$$P_{t+1} = 1 - (1 - P_t)^s + m(1 - P_t)^s. \tag{2}$$

3.3. Random migrants replace bad individuals: $(R \vdash B)$

When random migrants replace bad individuals, we need to know how many migrants are good. As migrants are chosen randomly, the proportion of good migrants is the same as the proportion of good individuals present in a deme, so $1 - (1 - P_t)^s$. Thus, the proportion of good individuals at the receiving deme is incremented by good migrants as

$$P_{t+1} = 1 - (1 - P_t)^s + m(1 - (1 - P_t)^s) = (1 + m)(1 - (1 - P_t)^s). \tag{3}$$

Note: Likewise when random migrants replace random individuals $(R \vdash R)$. This policy use random migrants for replacing random individuals. When we imagine that fitness at all demes is approximately the same, there is no effect on the takeover time t^*. As it was shown by published experimental results [2]. If it is not stated otherwise, the input parameters of the equations for the figures in the article are: Population size in a deme (n or Popsize) = 1000, tournament selection (TS) = 2 and migration rates (Migrate) = 0.05.

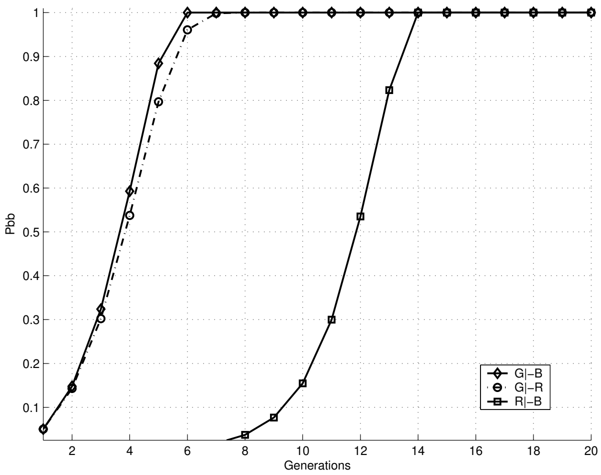


Figure 2: Figure shows the dependency of Pbb based on Generations (Time) for three replacement policies. The policies differ in the time convergency. The $R \vdash B$ policy was omitted.

4. Estimate of good individuals in a deme

We suppose that the estimate of good individuals in a deme at the start time ($t = 0$), which was defined as $P_0 = 1/n$, does not reflect the reality even for the Onemax function. As it is presented in 3, it should be changed because it does not reflect the reality. We changed it in such a way, that we run a simple random function, which gives P_0 more accurately. The random function depends on the test problem (Onemax) and PopSize n.

5. Derivation of takeover times

This section starts with definition of takeover times, then it clarifies what takeover times are and what they mean. *Takeover times* are times when a population gets uniform and does not improve it's fitness anymore. The uniformity is ensured by selection and migration operators. Those operators improve overall fitness in

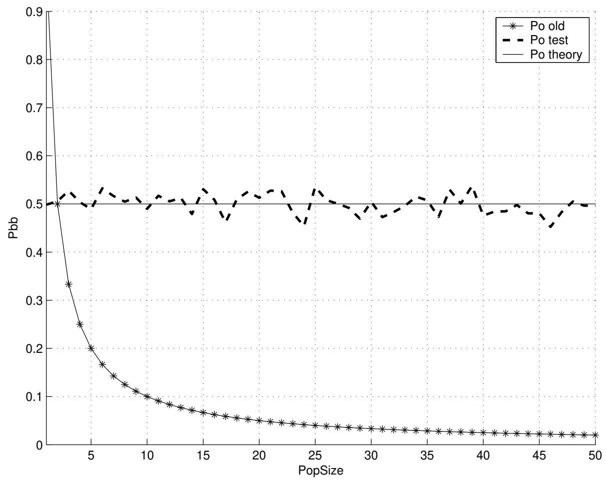


Figure 3: Figure describes the issue of P_0. There are three curves. The first one is for the old model. The second one reflects experimental data and the third one is the theoretical curve for Onemax function.

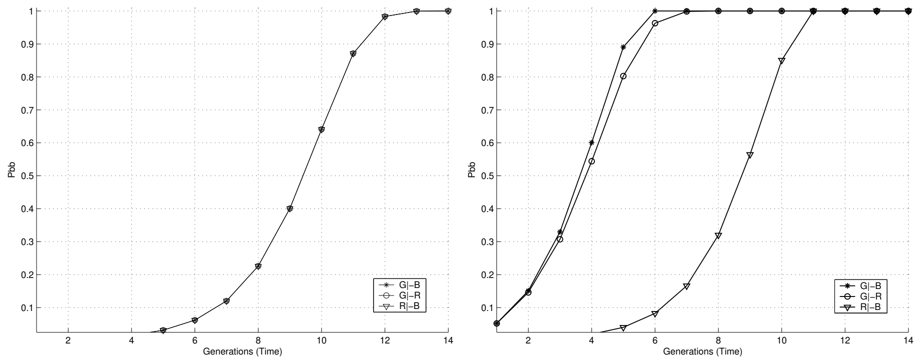


Figure 4: P_{bb} and migration rate in a deme. Figures show that migration rate has an impact on P_{bb} and the convergence time in a deme with P_0 based on the old estimate of P_0. Migration rate was set to $m = 0.00$ (Left) and migration rate was $m = 0.05$ (Right).

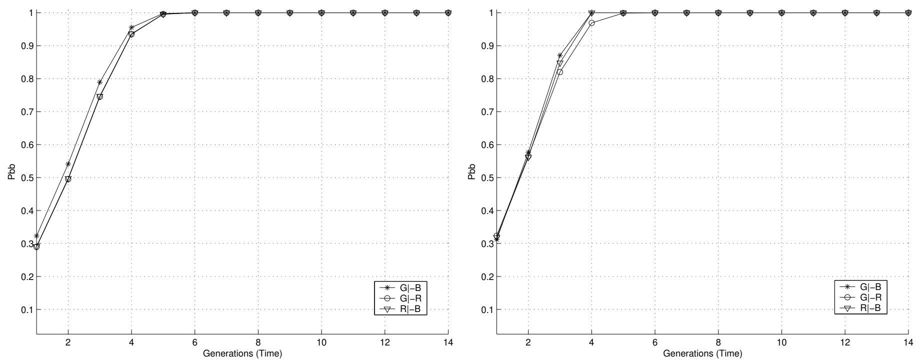


Figure 5: P_{bb} and migration rate in a deme. Figures show that migration rate has an impact on P_{bb} and the convergence time in a deme based on the new estimate of P_0. Migration rate was set to $m = 0.00$ (Left) and migration rate was $m = 0.05$ (Right).

the population by giving priority to better (good) individuals against worse (bad) ones (in our case, good and bad).

From the approximations of equations below, the takeover time t^* is gained in a such way, that P_t equals $\frac{n-1}{n}$ (all but one are good individuals) and P_0 equals $\frac{1}{n}$ (only one individual is good) and t^* is derived out.

5.1. Good migrants replace random individuals: $(G \vdash R)$

This is a new approximation of equation (2) to obtain the relevant takeover time. Important to note that the equation has changed a bit. P_t is not derived from a previous step P_{t-1} as in the equation (2), but it is from the starting point P_0. The equation is

$$P_t = 1 - (1 - P_0)^{s^{t^2}} (1 - m)^{s^{t^2}}. \tag{4}$$

5.2. Random migrants replace bad individuals: $(R \vdash B)$

To get the relevant take over time, this is a new approximation of equation (3). Similarly, as in the previous equation, P_t is not derived from a previous step P_{t-1} as in the equation (2), but it is from the starting point P_0. P_t is defined as

$$P_t = (1 - (1 - P_0)^{s^{t^2}})(1 + m). \tag{5}$$

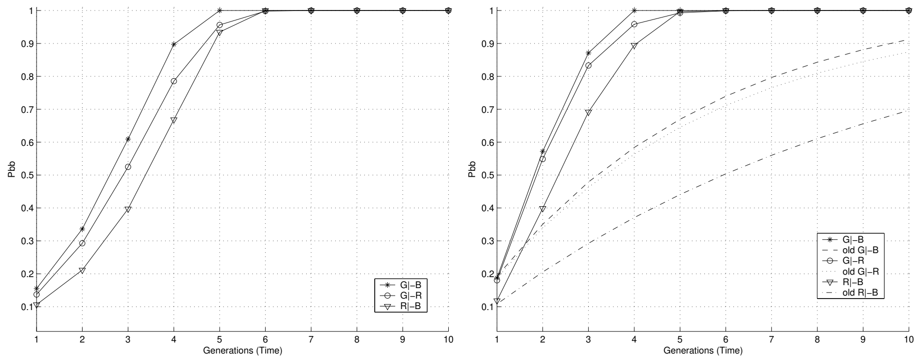


Figure 6: The recurrent equations ($P_t \rightarrow P_{t+1}$) vs. the approximation equations ($P_0 \rightarrow P_t$). The left one shows the recurrent equations. The right one presents new and old approximations for the approximation ones.

5.3. Takeover times

Equations of takeover times t^* were derived from the approximation equations. The data for $t^* : (R \vdash R)$ were derived from experimental measurements, because we did not have neither a recurrent equation nor an approximation for this policy.

Takeover time $t^* : (G \vdash B)$ is

$$t^* \approx -0.5(\frac{1}{\ln s} \ln \big(\frac{\ln(1-m) + \ln(1 + 1/n)}{\ln(1/n)}\big)). \tag{6}$$

Takeover time $t^* : (G \vdash R)$ is

$$t^* \approx -0.55(\frac{1}{\ln s} \ln \big(\frac{\ln(1-m) + \ln(1 + 1/n)}{\ln(1/n)}\big)). \tag{7}$$

Takeover time $t^* : (R \vdash B)$ is

$$t^* \approx 0.75\left(\frac{1}{\ln s}\left[\ln\left(\frac{\ln(\frac{m+1/n}{1+m})}{\ln(\frac{n-1}{n})}\right)\right]\right). \tag{8}$$

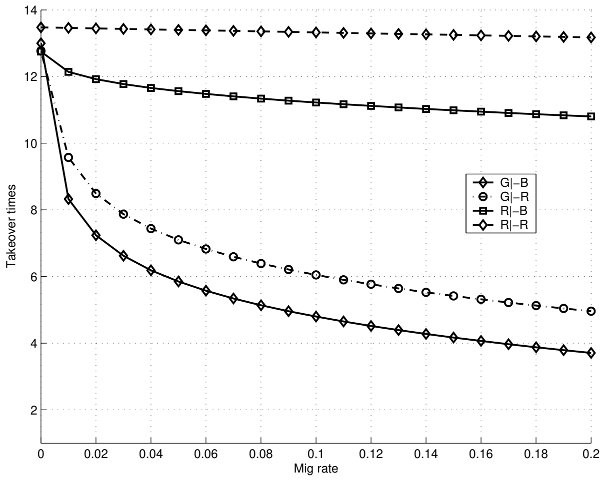


Figure 7: The diagram presents four replacement policies and their takeover times depending on the migration rate. Parameters are: Popsize = 1000, tournament selection = 2 and migration rates = $0.0 - 0.2$.

6. Conclusion

The modified time-convergence model of a deme shows many advantageous features for theoreticians as well as practitioners. It provides recurrent equations for computation of the proportion of the good individuals in a deme. From those equations, the approximation equations and later the equations for takeover times are derived. As one could see, there are some features in mentioned equations which were not tackled. There are described shortly in the following list and they are under our research.

- The model does not reflect the **problem difficulty**, so there is no relation to the difficulty of a problem being solved by a parallel genetic algorithm. The proposed model works only for Onemax function.
- The assumption is that **migration** must be a constant, therefore the model is too static. And it might have been more flexible in that view.
- **No bounds for migration** rate were specified. By this statement, we mention that there are no bounds on the size of migration rate, the number of neighbor and topology of demes.
- Currently, there is no equation for the $(R \vdash R)$ policy.

This work is a first part of the research on the convergence-time model of a deme in parallel genetic algorithms. Currently, in the second part, we focus on the above mentioned flaws to change the model in all mentioned "problem" points.

References

[1] D. E. Goldberg, K. Deb, "A comparative analysis of selection schemes used in genetic algorithms", *Foundation of Genetic Algorithms, 1*, pp. 69–93, 1991.

[2] E. Cantú-Paz, "*Migration Policies and Takeover Times in Parallel Genetic Algorithms*", IlliGAL Report No. 99008, p. 11, 1999.

[3] E. Cantú-Paz, "*Migration Policies, Selection Pressure, and Parallel Evolutionary Algorithms*", IlliGAL Report No. 99015, p. 20., 1999.

[4] Z. Konfršt, "Migration Policies and Takeover Times in PGAs", *APLIMAT'2003*, STU, Bratislava, pp. 447–453, 2003.

A Brief Comparison of Two Weighing Strategies for Random Forests

Post-Graduate Student:
ING. EMIL KOTRČ
Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2
182 07 Prague 8
kotrc@cs.cas.cz

Supervisor:
RNDR. PETR SAVICKÝ, CSC.
Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2
182 07 Prague 8
savicky@cs.cas.cz

Field of Study:
Mathematical Engineering
Classification: P3913

Abstract

Random Forests (RF) method is very successful and powerful classification (and regression) method based on ensembles of decision trees. To grow diverse trees in an ensemble RF uses two randomization techniques and all trees in an ensemble are created equal - trees gain the same weight. Although L. Breiman proved that RF converges and does not overfit, there exist some improvements of the basic classification algorithm. This paper is concerned with two improvements, which are based on weighing of individual trees. Some of the basic principles of these two approaches are very similar, but there are significant differences, which will be discussed in this paper.

1. Introduction

This paper deals with a brief theoretical comparison of two different weighing techniques for the Random Forests (RF) method [3]. The first technique is based on leaf confidences and it is described in [9]. The second is based on the analysis of the paper [8] and of the source codes developed by Marko Robnik[1].

At first we will briefly give some basic notions. We will be concerned with standard two-class classification problem. Let $\mathcal{X}$ be our domain space and let $\mathbf{x} = (x_1, \ldots, x_k, \ldots, x_n) \in \mathcal{X}$ be a vector of n measurements (attributes, variables). We will call these vectors as cases, instances, examples or similar. Let $\mathcal{C} = \{0, 1\}$ be the class label set of two classes 0 and 1. The class 1 can represent in applications some useful signal, and the class 0 noise or some background. We want to classify (predict) an unseen case $\mathbf{x}$ into the one of two classes using some classifier h. This classifier is a function $h : \mathcal{X} \to \mathcal{C}$, and it is build using some learning algorithm on some learning set. Learning (or training) set is the set of cases from the space $\mathcal{X}$ with the known class. Let $\mathcal{L} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ be our training set, where $\mathbf{x}_j \in \mathcal{X}$ is a case and y_j is its true class. Training cases with the class label equal to 1 are called positive examples, the others are called negative.

There exist several learning algorithms for building classifiers, see for example [4], but this paper is concerned with improvements of classifiers built by RF method. The next section briefly describes this method for growing decision forests. After that we will introduce two improvements of prediction step of this method and the closing section will be dedicated to the short summary and to the ideas for further work.

[1] `http://lkm.fri.uni-lj.si/rmarko`

2. Random Forests

Random Forests technique grows ensemble of decision trees (decision forest) as a classifier and it is described in [3]. Each tree in a forest is a standard binary decision tree (according to the CART methodology [1]) with two kinds of nodes - decision nodes and leaves. Decision nodes contain univariate tests, for example in the form $X_k \leq a$, where X_k is some of n attributes and a is some threshold. To classify using one decision tree, a case $\mathbf{x}$ starts its path in the root node of decision tree and it goes through the tests in decision nodes. If it satisfies a condition (test), it continues by the left way otherwise by the right way. This process is repeated until a leaf is encountered and the final prediction is given by that leaf.

Random Forests uses set of decision trees to make a prediction. To grow diverse decision trees RF uses two randomization techniques - bagging (bootstrap aggregation), see [2], and randomization of split selection. For each tree a new training set (bootstrap sample) is drawn at random from the original training set. Individual decision trees are grown on the new bootstrap sample using the CART methodology, see [1], with two differences - RF does not prune trees, it grows them to the maximal size and the tests in decision nodes are chosen using a simple randomization, for more details see [3].

Let us mention more details about bagging, since it will become important in the next sections. The bootstrap sample for each decision tree is drawn at random with replacement from the original learning set and it has the same size as the learning set, it means that a sample contains m cases. Because of drawing with replacement some cases (approximately $1/e \approx (1-1/m)^m$ %) do not occur in a sample. We will call these cases out of bag (OOB) for the given tree. On the other hand some cases may occur more than once in a sample, these will be called in bag cases. Let us have a simple example.

Example 3 *Let $\mathcal{L} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5\}$ be our learning set (classes y_i are omitted) and let $\mathcal{L}_1 = \{\mathbf{x}_1, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_2, \mathbf{x}_5\}$ be a random sample with replacement for the first tree. Then the cases $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_5$ are inbag, and the cases $\mathbf{x}_3$ and $\mathbf{x}_4$ are out of bag.*

RF uses a simple majority voting approach for classification. It means that the final prediction is given by the majority of votes of individual trees in the forest. For general problem we can define it as follows

$$h(\mathbf{x}) = C \text{ where } C = \arg\max_c |\{i \in \{1, \dots, N\} \mid h_i(\mathbf{x}) = c\}| \tag{1}$$

where $h_i(\mathbf{x})$ is the prediction given by i-th tree for a case $\mathbf{x}$. In this case, all trees have the same weight. Weighing strategies try to assign weights to trees or to leaves to improve the final prediction. This paper introduces two such strategies in the next two sections.

To close this section we will give some notions, which will be useful for the next sections. A forest will be a set of N decision trees $\mathcal{T} = \{T_1, \dots, T_N\}$ and $T_i(\mathbf{x})$ will denote a leaf which is encountered by the case $\mathbf{x}$ in the tree T_i. If u is a leaf, then $\mathbf{x} \in u$ means that a case $\mathbf{x}$ encountered leaf u.

3. Leaf Confidences

This first type of voting assigns weights (confidences) to the leaves of individual trees (not to the whole tree) during the learning phase. This approach is inspired by papers [5] and [10] and it is described in [9] in more details, this paper gives only a short summary.

It is very simple to define leaf confidences for two class problem. Let u stands for a leaf and $c(u) \in \mathcal{R}$ for its confidence. Positive values of leaf confidence $c(u)$ means preference for the class 1 and negative for the class 0. The level of confidence is given by the absolute value $|c(u)|$. The higher absolute value the higher confidence level.

Leaf confidences depend on the training set only. After a forest is grown all training cases are passed through each tree and each leaf u gains two statistics

$$\mathrm{pos}(u) = |\{\mathbf{x}_j \in \mathcal{L} \mid \mathbf{x}_j \in u,\, y_j = 1\}| \tag{2}$$

$$\mathrm{neg}(u) = |\{\mathbf{x}_j \in \mathcal{L} \mid \mathbf{x}_j \in u,\, y_j = 0\}| \tag{3}$$

$\mathrm{pos}(u)$ represents the number of positive training cases which reached the leaf u and $\mathrm{neg}(u)$ represents the number of negative training cases in the leaf u. We will use these two numbers to determine a confidence of a leaf. The larger and purer leaves will obtain higher confidence, the smaller leaves and leaves with nearly equal pos and neg will obtain smaller confidence. Let $w : \mathcal{N} \times \mathcal{N} \to R$ be a function which takes pos and neg as its arguments. Then we define leaf confidence as

$$c(u) = w(\mathrm{pos}(u), \mathrm{neg}(u)) \tag{4}$$

The study [9] was concerned with searching the most accurate leaf confidence. The best weights w were found using a simple statistical model, which described the behaviour of an ensemble grown by RF. We have tried several functions w, some of them were more successful some of them less. The simplest weights called rf simulate the true behaviour of RF - the majority voting.

$$rf(n_1, n_0) \stackrel{\mathrm{def}}{=} \mathrm{sign}(n_1 - n_0) \tag{5}$$

Instead of using all training cases in rf weights, RF uses only the inbag cases to assign a prediction to leaves. Since these weights reached quite good results we also used the sigmoidal function $s(x) = (1 - e^{-x})/(1 + e^{-x})$ to approximate the signum function in (5). These weights we denote as σ and they are parametrized by k. The definition of σ weights follows

$$\sigma_{(k)}(n_1, n_0) \stackrel{\mathrm{def}}{=} s\left(\frac{n_1 - n_0}{k \cdot (n_1 + n_0)}\right) \tag{6}$$

One of the best results in our experiments in previous work were reached by the simple weights called normalized difference (nd) defined as follows

$$nd_{(\alpha,h)}(n_1, n_0) \stackrel{\mathrm{def}}{=} \frac{n_1 - n_0 - h}{(n_1 + n_0)^{\alpha}} \tag{7}$$

These nd weights use two parameters h and α, the best results we got using $h = 0$ and $\alpha = 0.9$.

The final prediction of RF with leaf confidences is based on the sum of leaf confidences of leaves reached by the unknown case $\mathbf{x}$. Let

$$F(\mathbf{x}) = \sum_{i=1}^{N} c(T_i(\mathbf{x})) \tag{8}$$

be this sum. If $F(\mathbf{x})$ is greater than some threshold t, the final prediction is 1 otherwise 0. In other words, the final prediction is parameterized by t and t has a close relation to the desired background acceptance (probability that a case from the class 0 will be classified as 1). You can see the prediction function in the following definition

$$h(\mathbf{x}) = \begin{cases} 1 & \text{iff } F(\mathbf{x}) > t \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

The advantages of leaf confidences is their simplicity. Since leaf confidences can be assigned during the learning process, they do not slow down the prediction procedure. The best results were reached on smaller forests, as the size of forest (N) was growing the improvement was smaller. Leaf confidences are implemented using R statistical system [7], only two classes and numerical predictors are supported at present.

4. Margin weighing (weighted voting)

This section introduces much more complicated approach of weighing. All equations were derived from the original Robnik's source codes, since the description in the paper [8] is very poor. In spite of that more than two classes are supported by this type of voting, we will simplify it for two classes only to be able to make a comparison to leaf confidences. Margin weighing is based on the two step procedure using primary and secondary weights based on the margin of trees. During the learning process there are assigned primary weights to single leaves in each tree, similarly to the leaf confidences technique. The second step is taken in the prediction phase, where secondary weights are computed for each tree using margins of trees. Let us describe this process in more detail, for a short summary see the table 1.

Let T_i stands for a tree in a forest, where $i \in \{1, \dots, N\}$ and let $\mathbf{x}_j$ denotes a training case with the true class label $y_j \in \{0, 1\}$ for $j \in \{1, \dots, m\}$. Let us define $\operatorname{sign}^*(0) = -1$ and $\operatorname{sign}^*(1) = 1$. For each tree T_i and each training case $\mathbf{x}_j$ we can define the number $b_{i,j}$, which represents the number of occurrence of a case $\mathbf{x}_j$ in the inbag sample of the tree T_i.

Example 4 *For our example on page 59,* $b_{1,1} = 2$, $b_{1,2} = 2$, $b_{1,3} = 0$, $b_{1,4} = 0$ *and* $b_{1,5} = 1$.

Remember that for building each tree a new set of training cases is drawn at random with the same size as the original training set and that it is drawn with replacement! Then the out of bag set (OOB) for the i-th tree can be seen as the set

$$O_i = \{\mathbf{x}_j \in \mathcal{L} \mid b_{i,j} = 0\} \tag{10}$$

To use the margin weighing we will need the number of positive and negative cases in each leaf similarly to the leaf confidences in previous section, see (2) and (3). But here is first significant difference - for margin weighing we will compute pos and negs on the basis of the inbag sample not on the whole learning set! So let us define

$$\mathrm{pos}^*(u) = \sum_{y_j=1;\, T_i(x_j)=u} b_{i,j} \tag{11}$$

$$\mathrm{neg}^*(u) = \sum_{y_j=0;\, T_i(x_j)=u} b_{i,j} \tag{12}$$

for each leaf u in each tree T_i. It can be seen that $\mathrm{pos}^*(u)$ can be smaller than $\mathrm{pos}(u)$ if some positive example is missing in the bootstrap sample, and that $\mathrm{pos}^*(u)$ can be greater than $\mathrm{pos}(u)$ since positive examples may occur more than once in the inbag sample. We also have to mention that a lot of leaves will be pure on the inbag sample, since RF grows trees to pure or small leaves. So there will be a lot of couples $(\mathrm{pos}^*(u), \mathrm{neg}^*(u))$ with the zero on the one of these two positions.

Using these two statistics (11) and (12) we can define primary weights for margin weighing (in spite of that M. Robnik does not call it so). Primary weights can be assigned during the learning procedure as the leaf confidences. From the original Robnik's source codes we derived these weights in the form

$$c_1(u) = w(\mathrm{pos}^*(u), \mathrm{neg}^*(u)) \tag{13}$$

which is very similar to the leaf confidences approach, see (4). M. Robnik uses only one function w

$$w(n_1, n_0) = \frac{n_1 - n_0}{n_1 + n_0} \tag{14}$$

which is exactly equal to the $\mathrm{nd}_{(1,0)}$ weights, see (7). But still remember that this time we use inbag estimates of pos^* and neg^* and that is the significant difference. If the leaf u is pure on the inbag sample then $c_1(u) = \pm 1$. It can be easily seen that primary weights can be more generalised using some other function w, for example (5), (6) or (7).

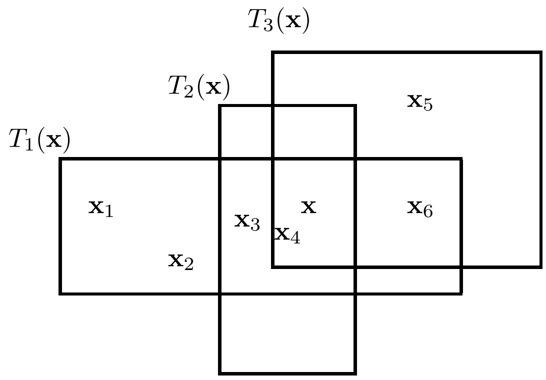


Figure 1: Three leaves reached by an unseen case $\mathbf{x}$ and by training cases $\mathbf{x}_1, \ldots, \mathbf{x}_6$

The second difference between margin weighing and leaf confidences is visible in the classification procedure. When leaf confidences are used, the prediction is defined upon the sum (8) of the leaf confidences of the leaves reached by an unknown case in each tree. The prediction using margin weights is much more difficult, since margin of each tree depends on all leaves reached by an unseen case $\mathbf{x}$. If we are going to classify a case $\mathbf{x}$ we will at first need the set of the "most" similar training cases. We will denote this set as $H(\mathbf{x})$. M. Robnik uses $H(\mathbf{x})$ equal to the set of τ training cases with the highest proximity to the case $\mathbf{x}$. The proximity will be defined bellow.

At first we have to find leave reached by the case $\mathbf{x}$ in each tree T_i. We will denote

$$H_i(\mathbf{x}) = \{\mathbf{x}_j \in \mathcal{L} \mid \mathbf{x}_j \in T_i(\mathbf{x})\}, \forall i \in \{1, \ldots, N\} \tag{15}$$

set of training cases covered by the leave $T_i(\mathbf{x})$. To select the most similar training cases to the unseen case $\mathbf{x}$ we will define the proximity of a training case first. The proximity of a training case $\mathbf{x}_j \in \bigcup_{i=1}^{N} H_i(\mathbf{x})$ (to the unseen case $\mathbf{x}$) is denoted as $\mathrm{prox}(\mathbf{x}_j)$ and it is defined as the number of occurrence in all leaves $T_i(\mathbf{x})$, in other words

$$\mathrm{prox}(\mathbf{x}_j) = \sum_{i=1}^{N} I(\mathbf{x}_j \in T_i(\mathbf{x})) \tag{16}$$

Where $I()$ is the indicator function, $I(true) = 1$ otherwise 0. The set of "similar" training instances can be the set of τ cases with the highest proximity. We will denote $H(\mathbf{x})$ as such set.

Example 5 *Figure 1 shows a simple example using only three trees. It depicts three leaves $T_1(\mathbf{x})$, $T_2(\mathbf{x})$ and $T_3(\mathbf{x})$ reached by a case $\mathbf{x}$ and it shows learning cases $\mathbf{x}_1, \ldots, \mathbf{x}_6$ covered by these leaves. Let sets H_i in this example be $H_1(\mathbf{x}) = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_6\}$, $H_2(\mathbf{x}) = \{\mathbf{x}_3, \mathbf{x}_4\}$ and $H_3(\mathbf{x}) = \{\mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6\}$. Let us take $\tau = 3$, then $H(\mathbf{x}) = \{\mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_6\}$, since $\mathrm{prox}(\mathbf{x}_4) = 3$, $\mathrm{prox}(\mathbf{x}_3) = \mathrm{prox}(\mathbf{x}_6) = 2$ and $\mathrm{prox}(\mathbf{x}_1) = \mathrm{prox}(\mathbf{x}_5) = \mathrm{prox}(\mathbf{x}_2) = 1$.*

As soon as we know the set $H(\mathbf{x})$ we are able to determine margins of each tree. The margin of the tree T_i depends on the out of bag cases of the tree T_i, which are similar to the case $\mathbf{x}$. More precisely, margin for the tree T_i will be defined using a set $S_i = H(\mathbf{x}) \cap O_i$. Let S_i be the set $S_i = \{\mathbf{x}_j^i \mid j = 1, \ldots, m_i\}$ then margin is defined as

$$\mathrm{mg}(T_i, S_i) = \frac{1}{|S_i|} \sum_{j=1}^{|S_i|} \mathrm{sign}^*(y_j^i) \cdot c_1(T_i(\mathbf{x}_j^i)) \tag{17}$$

Let $\mathbf{x}$ be an unseen case to classify and $T_1, \ldots, T_n$ be the forest

1. For each $i \in \{1, \ldots, N\}$ find leaves $T_i(\mathbf{x})$ reached by a case $\mathbf{x}$
2. For each $i \in \{1, \ldots, N\}$ find sets $H_i(\mathbf{x})$ of training cases covered by leaves $T_i(\mathbf{x})$
3. For each $\mathbf{x}_j \in \bigcup_{i=1}^{N} H_i(\mathbf{x})$ compute proximity $\mathrm{prox}(\mathbf{x}_j) = \sum_{i=1}^{N} I(\mathbf{x}_j \in T_i(\mathbf{x}))$
4. Let $H(\mathbf{x})$ contains τ cases $\mathbf{x}_j$ with the highest proximity
5. For each tree T_i, $i \in \{1, \ldots, N\}$, compute margin

$$\mathrm{mg}(T_i, S_i) = \frac{1}{|S_i|} \sum_{j=1}^{|S_i|} \mathrm{sign}^*(y_j^i) \cdot c_1(T_i(\mathbf{x}_j^i))$$

   where $S_i = H(\mathbf{x}) \cap O_i$, $\mathbf{x}_j^i \in S_i$

6. Final prediction is (9) using $F(\mathbf{x}) = \sum_{i=1}^{N} c_1(T_i(\mathbf{x})) \cdot c_2(T_i, S_i)$

Table 1: Prediction procedure using margin weights

The number $\mathrm{sign}^*(y_j^i) \cdot c_1(T_i(\mathbf{x}_j^i))$ will be positive if and only if the prediction of the i-th tree is correct, it means iff $h_i(\mathbf{x}_j^i) = y_j^i$. If the set S_i is empty, then the margin is set to 0.

The secondary weight is defined as the nonnegative part of the margin function

$$c_2(T_i, S_i) = \begin{cases} \mathrm{mg}(T_i, S_i) & \text{if } \mathrm{mg}(T_i, S_i) \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

You can easily see that the trees with the negative margin are left out from the prediction process for the case $\mathbf{x}$, since their secondary weight is set to zero. The final prediction is then defined as (9) using the weighted sum of primary weights

$$F(\mathbf{x}) = \sum_{i=1}^{N} c_1(T_i(\mathbf{x})) \cdot c_2(T_i, S_i) \tag{19}$$

The difference between this weighted sum (19) and the sum (8) is the usage of the secondary weights c_2. As written above primary weights (13) are in fact a special kind of leaf confidence (4) using the weights (7) with $\alpha = 1$ and $h = 0$. Leaf confidences and primary weights can be set during the training process. On the other hand the margins and secondary weights are set during the prediction phase, because they depend on the leaves, respectively on sets $H_i(\mathbf{x})$, of each tree in a forest. This process slows down and complicates significantly the prediction. Margin weighing is implemented as a part of standalone classification system and it supports more than two classes. At present we are going to implement it to the R system.

5. Remarks and Conclusion

The main contribution of this paper is to show main differences between two discussed weighing approaches and mainly to summarize the second approach - margin weighing. This paper is dedicated to the technical details only and it does not show any experimental comparison, but nevertheless we can see some significant differences and new ideas. For example the estimation using out of bag examples may be a good technique for determining pos and negs to compute leaf confidences. On the other hand we know that the function (14) does not have to be the best function, so there is a space to try some other which we have already used. A second problem using margin weights lies in the usage of the set S_i. It consists only of out of bag

examples similar to an unseen case, but it influences significantly the margin function. In other words, it is possible that the margin is computed using cases which are not covered by that leaf. Recall example 5 and figure 1, it can be possible that the margin of the leaf T_2 is computed using the cases $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_6$ and it is not what we would expect. So it is possible, that the margin is computed incorrectly and a "good" leaf is left out from the voting.
As you can see, the selection of most similar cases to compute margin can be very difficult and it brings some problems. It is true that RF and proximity is very close to the nearest neighbour method, see for example the paper [11], but margins bring complications by using out of bag cases. Our further work will be dedicated mainly to the more detailed and experimental comparison of two discussed weighing approaches and to the searching the new types of weighing strategies.

Acknowledgements
This work was partially supported by the Program "Information Society" under project 1ET100300517 and by the project MSM6840770010.

References

[1] Breiman L., Friedman J.H., Olshen R.A., Stone C.J., "Classification and regression trees", *Belmont CA:Wadsworth*, 1984

[2] Breiman L., "Bagging predictors", *Machine learning*, vol. 24, pp. 123–140, 1996

[3] Breiman L., "Random forests", *Machine learning*, vol. 45, pp. 5–32, 2001

[4] Hastie T., Tibshirani R., Friedman J. H., "The Elements of Statistical learning", *Springer-Verlag*, 2001

[5] Quinlan J.R., "Bagging, boosting and C4.5", *Proceedings of the Thirteenth National Conference in Artificial Intelligence*, pp. 725–730, 1996

[6] Quinlan J.R., "C4.5: Programs for Machine Learning", *Morgan Kaufmann*, 1992

[7] R Development Core Team, "R: A language and environment for statistical computing", *R Foundation for Statistical Computing*, www.r-project.org, 2004

[8] Robnik, M. "Improving random forests" *Machine learning, ECML proceedings*, 2004.

[9] Savický P., Kotrč E., "Experimental study of leaf confidences for Random Forests", *Proceedings of COMPSTAT*, 2004

[10] Shapire R.E., Singer Y. "Improved boosting algorithms using confidence rated predictions", *Machine learning*, vol. 37, pp. 297–336, 1999

[11] Yi Lin, Yongho Jeon, "Random Forest and Adaptive Nearest Neighbors", *Department of Statistics*, Technical Report No.1055, 2002

Kernel Based Regularization Networks

Post-Graduate Student:
RNDR. PETRA KUDOVÁ
Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2
182 07 Prague 8
petra@cs.cas.cz

Supervisor:
MGR. ROMAN NERUDA, CSC.
Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2
182 07 Prague 8
roman@cs.cas.cz

Field of Study:
Theoretical Computer Science
Classification: I1

This work was supported by GA ČR grant 201/05/0557 and by the Institutional Research Plan AV0Z10300504 "Computer Science for the Information Society: Models, Algorithms, Appplications".

Abstract

We discuss one approach to the problem of *learning from examples* – the Kernel Based Regularization Networks. We work with a learning algorithm introduced in [1]. We will shortly introduce special types of kernels: *Product* kernels and *Sum* kernels (joint work with T. Šámalová). We will describe technique we use to estimate the explicit parameters of the learning algorithm, the regularization parameter and the type of kernel. The performance of described algorithms will be demonstrated on experiments, including a comparison of different types of kernels.

1. Introduction

The problem of *learning from examples* (also called *supervised learning*) is a subject of great interest. The need for a good supervised learning technique stems from a wide range of application areas, covering various approximation, classification, and prediction tasks.

Kernel methods represent a modern family of learning algorithms. A comprehensive overview of kernel learning algorithms can be found in [2, 3, 4].

In this work we study one type of a kernel based learning algorithm, Regularization Network, a feed-forward neural network with one hidden layer, derived from the regularization theory. While it is well studied from the mathematical point of view, we are more interested in practical points of its application.

In the following two sections we introduce Regularization Network with its basic learning algorithm. In section 4 we present two special types of kernels, Product and Sum kernel (join work with T. Šámalová), section 5 deals with a choice of a kernel type and the a regulatization parameter. Experimental results are reported in section 6.

2. Regularization Network

Our problem can be formulated as follows. We are given a set of examples (pairs) $\{(\vec{x}^i, y^i) \in R^d \times R\}_{i=1}^N$ that was obtained by random sampling of some real function f, generally in the presence of noise (see 1). To this set we refer as *a training set*. Our goal is to recover the function f from data, or to find the best estimate of it. It is not necessary that the function exactly interpolates all the given data points, but we need

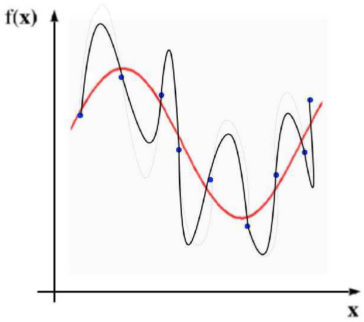


Figure 1: The problem of learning from examples

a function with a good *generalization*, that is a function that gives relevant outputs also for the data not included in the training set.

This problem is generally ill-posed, there are many functions interpolating the given data points, but not all of them exhibit also the generalization ability.

Therefore we have to consider some a priori knowledge about the function. It is usually assumed that the function is *smooth*, in the sense that two similar inputs corresponds to two similar outputs and the function does not oscillate too much. This is the main idea of the regularization theory, where the solution is found by minimizing the functional (1) containing both the data term and the smoothness information.

$$H[f] = \frac{1}{N}\sum_{i=1}^{N}(f(\vec{x}^i) - y^i)^2 + \gamma\Phi[f], \tag{1}$$

where Φ is called a *stabilizer* and $\gamma > 0$ is *the regularization parameter* controlling the trade-off between the closeness to data and the smoothness of the solution.

Poggio and Smale in [1] studied the Regularization Networks derived using a Reproducing Kernel Hilbert Space (RKHS) as the hypothesis space. Let $\mathcal{H}_K$ be an RKHS defined by a symmetric, positive-definite kernel function $K_{\vec{x}}(\vec{x}') = K(\vec{x}, \vec{x}')$. Then if we define the stabilizer by means of norm $||\cdot||_K$ in $\mathcal{H}_K$ and minimize the functional

$$\min_{f\in\mathcal{H}_K} H[f], \qquad \text{where } H[f] = \frac{1}{N}\sum_{i=1}^{N}(y^i - f(\vec{x}^i))^2 + \gamma||f||^2_K \tag{2}$$

over the hypothesis space $\mathcal{H}_K$, the solution of minimization (2) is unique and has the form

$$f(\vec{x}) = \sum_{i=1}^{N} w_i K_{\vec{x}^i}(\vec{x}), \qquad (N\gamma I + K)\vec{w} = \vec{y}, \tag{3}$$

where I is the identity matrix, K is the matrix $K_{i,j} = K(\vec{x^i}, \vec{x^j})$, and $\vec{y} = (y^1, \ldots, y^N)$.

The solution (3) can be represented by a feed-forward neural network with one hidden layer and a linear ouput layer (see (2)). We refer to a network of this form as *Regularization Network* (RN). The units of hidden layer realize the *kernel function* $K(\cdot,\cdot)$, with the first argument fixed to $\vec{c}_i$. We will refer to the vectors $\vec{c}_i$ as *centers*. In the optimal solution (3), they are fixed to the data points $\vec{x}^i$.

3. RN learning algorithm

The Regularization Network scheme derived in previous section leads to the RN learning algorithm (Algorithm 3.1).

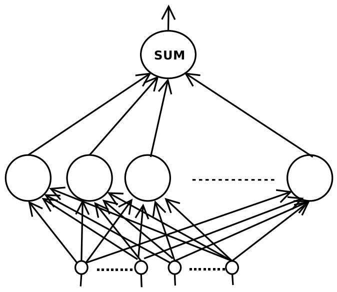


Figure 2: Regularization Network Scheme.

The algorithm is simple and effective for data sets of small and medium size, i.e. those for which we can solve the linear system (4) by means of available numerical software using current computational resources.

The tasks with huge data sets are harder to solve using this simple algorithm, and they lead to solutions of implausible size as well. Other algorithms should be used in such cases, RBF networks represent one option. They belong to the family of Generalized Regularization Networks, see [5] for a comparison of RBF networks and Regularization Networks. Alternatively, in the next section we propose the *Divide et Impera* approach, dividing the task to several smaller subtasks.

The strength of the algorithm stems from the fact, that the linear system we are solving is well-posed, i.e. it has a unique solution and the solution exists. (It has N variables, N equations, K is positive-definite and $(N\gamma I + K)$ is strictly positive.) We are interested in its real performance in the first place. Therefore we are also asking the question, if the system is well-conditioned, i.e. insensitive to small perturbations of the data.

The rough estimate of well-conditioning is the condition number of the matrix $(N\gamma I + K)$. [6] shows that the condition number depands only on the regularization parameter and the separation radius (the smallest distance between two data points). We know that the condition number is small, if $N\gamma$ is large, i.e. the matrix has a dominant diagonal. Unfortunately, we are not entirely free to choose γ, because with too large γ we loose the closeness to data. See figure 3b.

Both the numerical stability and the real performance of the algorithm depends significantly on the choice of the regularization parameter γ and the type of kernel (see 3a). These parameters are explicit parameters of the algorithm and their optimal choice always depends on the particular task. We will discuss the techniques we use to estimate these parameters in the section 5 .

Input: Data set $\{\vec{x}^i, y^i\}_{i=1}^N \subseteq \mathcal{R}^n \times \mathcal{R}$
Output: Regularization Network.

1. Set the centers of kernels:

$$\forall i \in \{1, \ldots, N\} : \vec{c}^i \leftarrow \vec{x}^i$$

2. Compute the values of weights $w_1, \ldots, w_N$:

$$(N\gamma I + K)\vec{w} = \vec{y}, \tag{4}$$

where I is the identity matrix,
$K_{i,j} = K_1(\vec{c}^i, \vec{x}^j)$,
and $\vec{y} = (y^1, \ldots, y^N)$, $\gamma > 0$.

Algorithm 3.1. RN learning algorithm

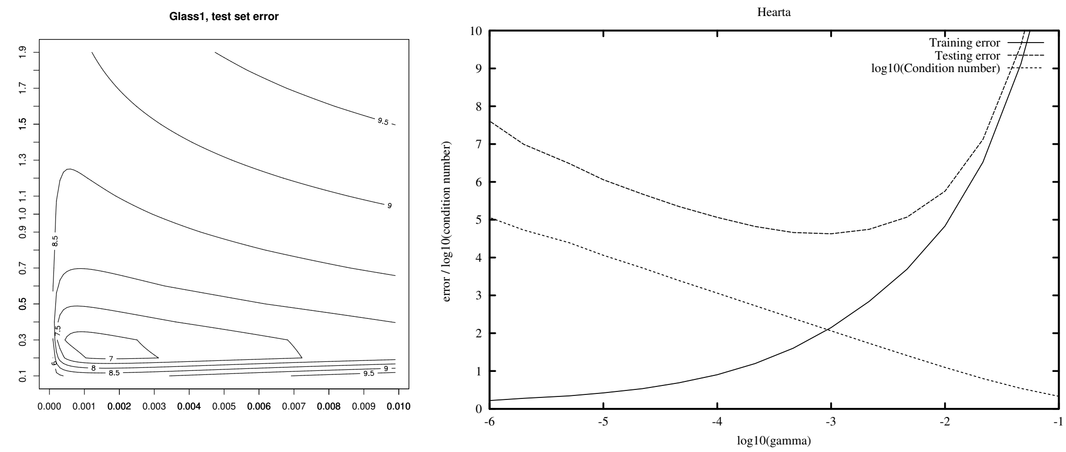


Figure 3: a) The dependence of the error function (computed on test set) on parameters γ a b (the width of the Gaussian kernel). b) The relation between γ and training and testing error.

4. Product and Sum Kernels

In [7](joint work with T. Šámalová) we proposed two types of composite kernels, Product kernels and Sum kernels. These kernels should better reflect the character of data.

By a *Product kernel* (see fig 4b) we mean an unit with $(n+m)$ real inputs and one real output. It consists of two positive definite kernel functions $K_1(\vec{c_1},\cdot)$, $K_2(\vec{c_2},\cdot)$, one evaluating the first n inputs and one evaluating the other m inputs. The output of the product unit is computed as the product $K_1(\vec{c_1},\vec{x_1}) \cdot K_2(\vec{c_2},\vec{x_2})$.

Product kernels are supposed to be used on data with different types of attributes, since different groups of attributes can be processed by different kernel functions. See [8].

By a *Sum kernel* (see fig 4a) we mean a unit with n real inputs and one real output. It consists of two positive definite kernel functions $K_1(\vec{c},\cdot)$, $K_2(\vec{c},\cdot)$, both evaluating the same input vector. The output of the sum unit is computed as the sum $K_1(\vec{c},\vec{x}) + K_2(\vec{c},\vec{x})$.

There are two different motivations for the choice of Sum kernel.

The former supposes we have some a-priori knowledge of data suggesting that the solution is in a form of sum of two functions. The kernel is than a sum of two kernel functions , for instance two Gaussian functions of different widths

$$K(\vec{x},\vec{y}) = K_1(\vec{x},\vec{y}) + K_2(\vec{x},\vec{y}) = e^{-(\frac{\|\vec{x}-\vec{y}\|}{d_1})^2} + e^{-(\frac{\|\vec{x}-\vec{y}\|}{d_2})^2}. \tag{5}$$

The solution in this case has a form

$$f(\vec{x}) = \sum_{i=1}^{N} w_i \left(e^{-(\frac{\|\vec{x}-\vec{c_i}\|}{d_1})^2} + e^{-(\frac{\|\vec{x}-\vec{c_i}\|}{d_2})^2} \right) . \tag{6}$$

The latter supposes we have data with different distribution in different parts of the input space. Here we may want to let different kernels operate on different parts of the input space.

[7] shows that if K is a kernel function for an RKHS F, then function

$$K_A(\vec{x},\vec{y}) = \begin{cases} K(\vec{x},\vec{y}) & \text{if } \vec{x},\vec{y} \in A, \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

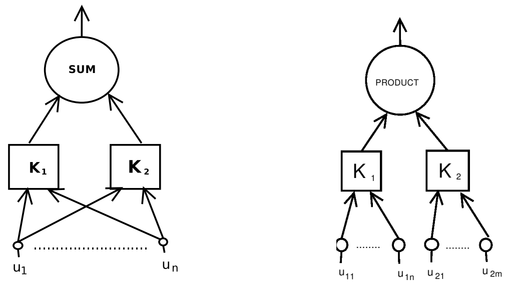


Figure 4: a) Sum Unit, b) Product Unit.

is a kernel function for the RKHS $F_A = \{f_A, f \in F\}$, where $f_A(\vec{x}) = f(\vec{x})$ if $\vec{x} \in A$ and $f_A(\vec{x}) = 0$ otherwise.

Suppose that we have a partition of the training set to k disjunct subsets A_i and for each subset we have a kernel function K_i. We can define

$$K(\vec{x}, \vec{y}) = \begin{cases} K_i(\vec{x}, \vec{y}) & \vec{x}, \vec{y} \in A_i \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

and we get a solution in the form

$$f(\vec{x}) = \sum_{i=1}^{N} w_i K(\vec{c_i}, \vec{x}) = \sum_{i \in I_1} w_i K_i(\vec{c_i}, \vec{x}) + \ldots + \sum_{i \in I_k} w_i K_k(\vec{c_i}, \vec{x}), \tag{9}$$

where $I_j = \{i, \vec{x}_i \in A_j\}$.

The formula (9) is also an justification for our *Divide et impera approach*. In case of huge data sets, we can divide the data set to several subtasks and apply the algorithm 3.1 on each of these subtasks, possibly in parallel. The solution is then obtained as a sum of the solutions of subtasks.

5. Choice of kernel function and regularization parameter

As we showed in the section 3, the choice of the explicit parameters (γ and the type of kernel) are crucial for the successful application of the Algorithm 3.1. There exist no easy way for estimation of these parameters, usually some kind of an exhaustive search for parameters with the lowest cross-validation error is used. The choice of regularization parameter was discussed for example in [9].

First we need a measure of a real performance of the network, that enables us to say that one particular choice of parameters is better than another. We use k-fold cross-validation to estimate a real performance of the network and its generalization ability.

Suppose we are given a training set of data $TS = \{\vec{x}^i, y^i\}_{i=1}^N \subseteq \mathcal{R}^n \times \mathcal{R}$. We split this data set randomly into k folds $TS_1, \ldots, TS_k$ such as $\bigcup_{i=1}^k TS_i = TS$ and $TS_i \bigcap_{i \neq j} TS_j \neq 0$. Then f_i is the network obtained by the Algorithm 3.1 run on the data set $\bigcup_{j \neq i} TS_j$. The cross-validation error is given by

$$E_{cross} = \frac{1}{k} \sum_{i=1}^{k} \frac{1}{|TS_i|} \sum_{(\vec{x}, y) \in TS_i} (f_i(\vec{x}) - y)^2. \tag{10}$$

In our experiments we use Gaussian kernels, so the choice of a kernel type is reduced to the choice of Gaussian width. We use the *Adaptive grid search* (Algorithm 5.1) for estimation of the regularization parameter

Input: Data set $\{\vec{x}^i, y^i\}_{i=1}^N \subseteq \mathcal{R}^n \times \mathcal{R}$
Output: Parameters γ and a width b.

1. Create a set of couples $\{[\gamma, b]_i, i = 1, \ldots, m\}$, uniformly distributed in $< \gamma_{min}, \gamma_{max} > \times < b_{min}, b_{max} >$.
2. For each $[\gamma, b]_i$ for $i = 1, \ldots, m$ and for each couple evaluate the cross-validation error (10) E^i_{cross}.
3. Select the i with the lowest E^i_{cross}.
4. If the couple $[\gamma, b]_i$ is at the border of the grid, move the grid. (see Fig. 5a)
5. If the couple $[\gamma, b]_i$ is inside the grid, create finer grid around this couple. (see Fig. 5b)
6. Go to 2 and iterate until cross-validation error stops decreasing.

Algorithm 5.1. Adaptive grid search

and the width of Gaussian kernels. The character of dependency of the error function on these parameters, as shown on Fig. 3a, enables us to start with a coarse grid and then create a finer grid around the point with the lowest cross-validation error.

The winning values of parameters found by the Algorithm 5.1 are then used to run the Algorithm 3.1 on the whole training set.

6. Experiments

The goal of our experiments was to demonstrate the performance of Regularization Networks and compare different kinds of kernels (Gaussian kernels, Product kernels and Sum kernels).

Gaussian kernels were used, regularization parameter and width were estimated by the Adaptive grid search (Alg. 5.1).

All algorithms are implemented in Bang [10], the standard numerical library LAPACK [11] was used for linear system solving.

Benchmark data repository Proben1 (see [12]) containing both approximation and classification tasks was

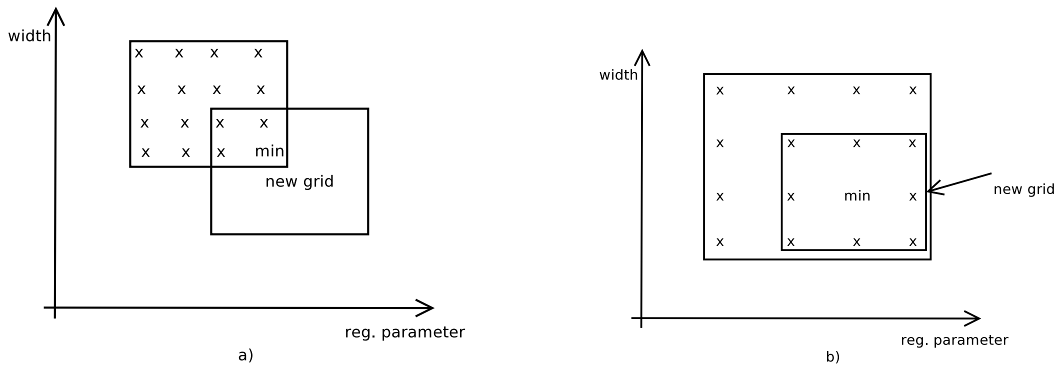


Figure 5: a) Move grid b) Create finer grid

chosen. The short description of Proben1 tasks is listed in table 1. Each task is present in three variants, three different partitioning into training and testing data.

Task name	n	m	N_{train}	N_{test}	**Type**
cancer	9	2	525	174	class
card	51	2	518	172	class
flare	24	3	800	266	approx
glass	9	6	161	53	class
heartac	35	1	228	75	approx
hearta	35	1	690	230	approx
heartc	35	2	228	75	class
heart	35	2	690	230	class
horse	58	3	273	91	class
soybean	82	19	513	170	class

Table 1: Overview of Proben1 tasks. Number of inputs (n), number of outputs (m), number of samples in training and testing sets (N_{train},N_{test}). Type of task: approximation or classification.

For each experiment we used separate data sets for training and testing. The normalized error was evaluated

$$E = 100\frac{1}{N}\sum_{i=1}^{N}||\vec{y}^i - f(\vec{x}^i)||^2,$$

where N is a number of examples and f is the network output.

In table 2 errors on training and testing set for Regularization network with Gaussian kernels (RN), Sum kernels and Product kernels are listed, together with number of evaluations needed to estimate the explicit parameters. Error values obtained by RBF are added to compare the performance of Regularization Networks with other standard neural network approach.

At Fig. 6 a kernel consisting of sum of two Gaussians found by parameter search is shown. This kernel showed an interesting behavior on several data sets. The error on the training set is almost a zero (rounded to zero) and still the generalization ability of the network is good, i.e. the error on testing set is not high. This is caused by the fact, that the kernel consists of two Gaussians, one being very narrow. The diagonal in matrix K from (4) is dominant and so regularization member is not needed, precisely γ is near to zero.

The figures 7,8 show the results obtained with proposed *Divide et impera* approach. We can see that this approach significantly reduces the time requirements, but we have to pay for it but slightly worse performance.

7. Conclusion

We discussed Kernel Based Regularization Network learning technique and special types of kernels, Sum and Product kernels.

We showed that the performance of basic RN algorithm depends on the choice of explicit parameters (kernel type, reg. parameter). We proposed a technique for estimation of these parameters, the Adaptive grid search.

On experiments we demonstrated the performance of RN algorithm with parameters estimated by the Adaptive grid search and compared performance of RN with classical kernels, Product kernels and Sum kernels.

By parameter search for Sum kernels we found an interesting type of kernel, that exhibit a good generalization on many tasks, even without regularization member.

Task	RN			Sum kernels			Product kernels			RBF	
Task	E_{train}	E_{test}	evals	E_{train}	E_{test}	evals	E_{train}	E_{test}	evals	E_{train}	E_{test}
cancer1	2.28	**1.75**	96	0.00	1.77	664	2.68	1.81	396	2.31	2.11
cancer2	1.86	3.01	76	0.00	**2.96**	624	2.07	3.61	412	1.91	3.12
cancer3	2.11	2.79	97	0.00	**2.73**	292	2.28	2.81	415	1.66	3.19
card1	8.75	10.01	126	8.81	10.03	472	9.22	**9.99**	6618	8.12	10.16
card2	7.55	**12.53**	101	0.00	12.54	376	7.96	12.90	6925	8.05	12.81
card3	6.52	12.35	113	6.55	12.32	387	6.94	12.23	6930	6.77	**12.09**
flare1	0.36	0.55	76	0.35	0.54	200	0.36	**0.54**	1008	0.37	**0.37**
flare2	0.42	0.28	60	0.44	**0.26**	164	0.42	0.28	756	0.41	0.31
flare3	0.38	0.35	36	0.42	**0.33**	164	0.40	0.35	1092	0.37	0.38
glass1	3.37	6.99	165	2.35	**6.15**	439	2.64	7.31	567	5.10	6.76
glass2	4.32	7.93	137	1.09	**6.97**	699	2.55	7.46	667	4.93	7.96
glass3	3.96	7.25	72	3.04	**6.29**	724	3.31	7.26	424	5.80	8.06
heart1	9.61	**13.66**	57	0.00	13.91	600	9.56	13.67	472	9.96	14.05
heart2	9.33	13.83	57	0.00	13.82	260	9.43	13.86	503	6.36	**11.67**
heart3	9.23	15.99	117	0.00	15.94	324	9.15	16.06	487	6.95	**12.02**
hearta1	3.42	4.38	134	0.00	4.37	532	3.47	4.39	1175	3.08	**4.36**
hearta2	3.54	4.07	134	3.51	4.06	478	3.28	4.29	476	3.36	**4.05**
hearta3	3.44	4.43	122	0.00	4.49	372	3.40	4.44	514	3.19	**4.29**
heartac1	4.22	**2.76**	496	0.00	3.26	483	4.22	**2.76**	1944	2.26	3.69
heartac2	3.50	3.86	342	0.00	**3.85**	500	3.49	3.87	1346	1.78	4.98
heartac3	3.36	**5.01**	405	3.36	**5.01**	1588	3.26	5.18	200	1.66	5.81
heartc1	9.99	16.07	900	0.00	**15.69**	470	10.00	16.08	3528	6.07	16.17
heartc2	12.70	**6.13**	917	0.00	6.33	680	12.37	6.29	3375	7.99	6.49
heartc3	8.79	12.68	121	0.00	**12.38**	760	8.71	12.65	496	7.13	14.35
horse1	7.35	**11.90**	121	0.20	**11.90**	408	14.25	12.45	1644	10.57	11.96
horse2	7.97	15.14	117	2.84	**15.11**	768	12.24	15.97	1332	10.04	16.80
horse3	4.26	**13.61**	85	0.18	14.13	328	9.63	15.88	479	9.88	14.56
soybean1	0.12	**0.66**	57	0.11	**0.66**	367	0.13	0.86	351	0.28	0.73
soybean2	0.24	**0.50**	85	0.25	0.53	175	0.23	0.71	463	0.38	0.60
soybean3	0.23	0.58	89	0.22	**0.57**	367	0.21	0.78	367	0.31	0.72

Table 2: Comparisons of errors on training and testing set for RN with Gaussian kernels, Sum kernels, Product kernels and RBF. The lowest testing error for each task is highlighted.

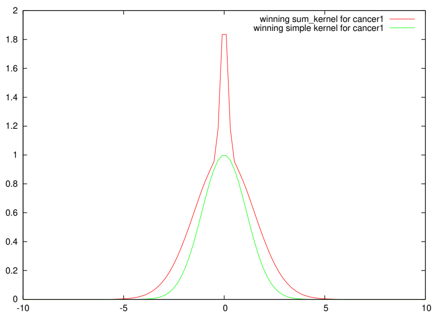


Figure 6: Kernels found by parameter search for cancer1 data set.

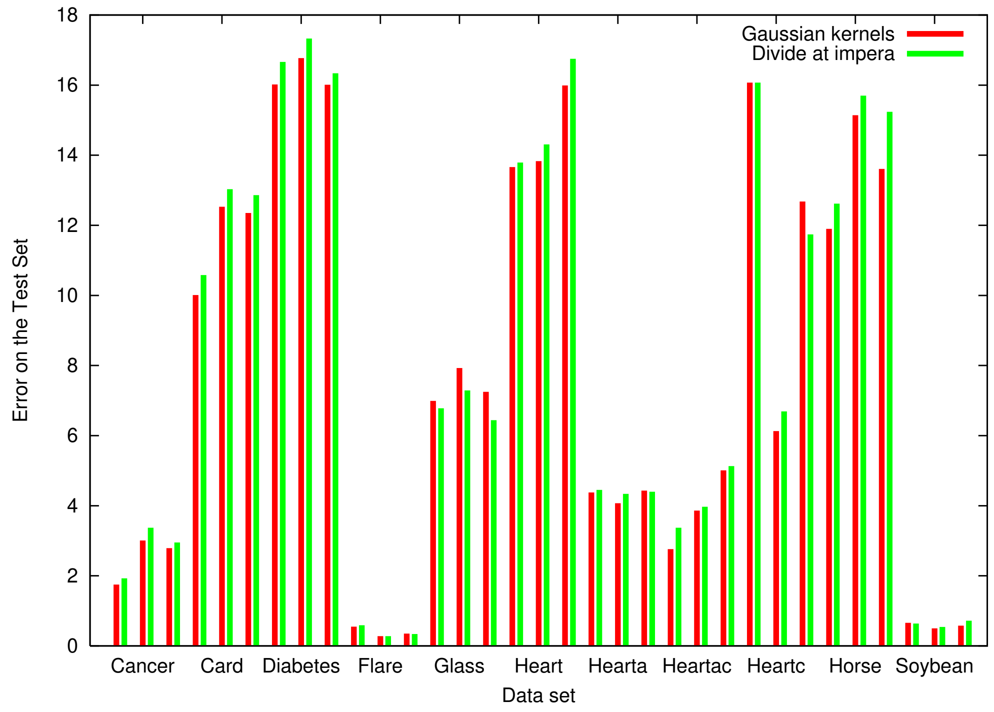


Figure 7: Comparison of the error on the testing set for RN with Gaussian kernels and Divide et impera approach.

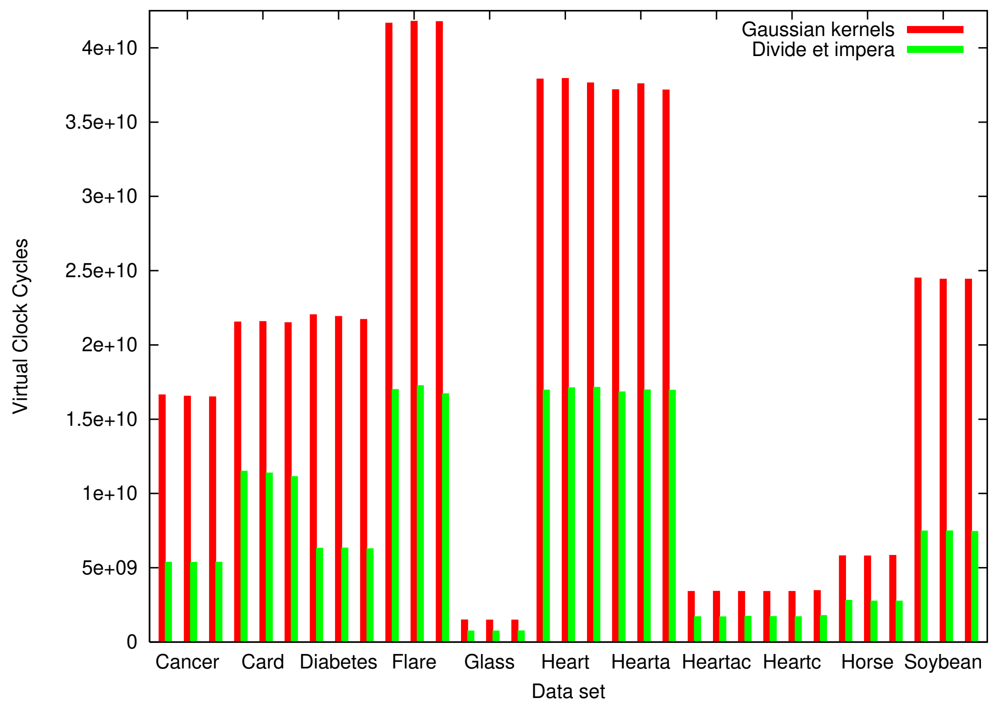


Figure 8: Comparison of time requirements of RN with Gaussian kernels and Divide et impera approach. Measured in clock cycles using [13].

References

[1] T. Poggio and S. Smale, "The mathematics of learning: Dealing with data," *Notices of the AMS*, vol. 50, pp. 536–544, 5 2003.

[2] B. Schoelkopf and A. J. Smola, *Learning with Kernels*. MIT Press, Cambridge, Massachusetts, 2002.

[3] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[4] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and Neural Networks architectures," *Neural Computation*, vol. 2, pp. 219–269, 7 1995.

[5] P. Kudová, "Comparison of kernel based regularization networks and RBF networks.," in *Proceedings of ITAT, 2004.*, 2004.

[6] F. Narcowich, N. Sivakumar, and J. Ward, "On condition numbers associated with radial-function interpolation," *Journal of Mathematical Analysis and Applications*, vol. 186, pp. 457–485, 1994.

[7] T. Šámalová and P. Kudová, "Sum and product kernel networks," Tech. Rep. 935, Institute of Computer Science, AS CR, 2005.

[8] P. Kudová and Šámalová T., "Product kernel regularization networks," in *Proceedings of ICANNGA, 2005*, Springer-Verlag, 2005.

[9] S. Haykin, *Neural Networks: a comprehensive foundation*. Tom Robins, 2nd ed., 1999.

[10] BANG, "Multi-agent system for experimentints with computational intelligence models.." `http://bang.sf.net/`.

[11] LAPACK, "Linear algebra package," `http://www.netlib.org/lapack/`.

[12] L. Prechelt, "PROBEN1 – a set of benchmarks and benchmarking rules for neural network training algorithms," Tech. Rep. 21/94, Universitaet Karlsruhe, 9 1994.

[13] PAPI, "Performance application programming interface," `http://icl.cs.utk.edu/papi/`.

Flows of Fluids with Pressure Dependent Viscosity in the Journal Bearing

Post-Graduate Student:
MGR. MARTIN LANZENDÖRFER

Mathematical Institute
Charles University
Sokolovská 83
CZ-186 75 Prague 8 - Karlín
Czech Republic;

Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2
182 07 Prague 8

lanz@karlin.mff.cuni.cz, lanz@cs.cas.cz

Supervisor:
DOC. RNDR. JOSEF MÁLEK, CSC.

Mathematical Institute
Charles University,
Sokolovská 83
CZ-186 75 Prague 8 - Karlín

Czech Republic

malek@karlin.mff.cuni.cz

Field of Study:
Mathematical modeling
Classification: F11

Abstract

Journal bearings that have been used for thousands of years and that go along with our civilization as well as the wheel, could be imagined as two eccentric cylinders, separated by fluid. Within this simple geometry we investigate the flow of non-Newtonian fluid.

In the first part, we describe the geometry, the fluid model, we briefly mention related theoretical results and previous investigations. In the second part, we provide numerical simulations of the planar steady flow within the annular region using the finite element method. We compare the classical Navier-Stokes model and the generalized one and provide several example simulations discussing the parameters of the model.

1. Introduction[1]

Lubrication generally, and the journal bearings as well, have been helping mankind for thousands of years. Basic laws of friction were first correctly deduced by da Vinci (1519), who was interested in the music made by the friction of the heavenly spheres. The scientific study of lubrication began with Rayleigh, who, together with Stokes, discussed the feasibility of a theoretical treatment of film lubrication. The journal bearings are heavily used in these days, and they are designed and studied on the mathematical basis and by numerical computations for a long time. Even by browsing the Internet you can find web sites where simple computational simulations are provided by an automatic software for free. (Mostly based on the Reynolds approximation.) This paper does not aspire to present any kind of directly applicable numerical result or method at all. The intentions of this work are rather to follow one of the lines of today's investigation; to present, in the perspective of numerical results, one of the recent generalizations of the Navier-Stokes model of fluid motion in the context of journal bearing lubrication problem: the main aim is to follow the theoretical results achieved in [10] and to study the capabilities of the constitutive model, for which we know that our problem formulation has a solution.

[1] Many of what is written in the beginning of this section can be found in [4].

Lubrication is used to reduce/prevent wear and lower friction. The behavior of sliding surfaces is strongly modified with the introduction of a lubricant between them. When the minimum film thickness exceeds, say, 2.5μm, the coefficient of friction is small, and depends on no other material property of the lubricant than its viscosity. When there is a continuous fluid film separating the solid surfaces we speak of fluid film bearings. Here we deal with the *self-acting* bearing, operating in the *hydrodynamical mode* of lubrication, where the film is generated and maintained by the viscous drag of the surface themselves, as they are sliding relative to one another. Hydrodynamic bearings vary enormously both in their size and in the load they support, from the bearings used by the jeweler, to the journal bearings of a large turbine generator set, which might be 0.8m in diameter and carry a specific load of 3MPa, or the journal bearing of a rolling mill, for which a specific load of 30MPa is not uncommon.

If the motion which the bearing must accommodate is rotational and the load vector is perpendicular to the axis of rotation, the hydrodynamic bearing employed is *journal bearing*. In their simplest form, a journal and its bearing consist of two eccentric, rigid, cylinders. The outer cylinder (bearing) is usually held stationary while the inner cylinder (journal) is made to rotate at an angular velocity ω. If the bearing is "infinitely" long, there is no pressure relief in the axial direction. Axial flow is therefore absent and changes in shear flow must be balanced by changes in circumferential pressure flow alone. In this paper, we follow this assumption, which allow us to restrict our further considerations to two-dimensional plane perpendicular to the axial direction.

We thus consider the geometry as it can be seen in figure 1. The domain of the flow is an eccentric annular ring, the outer circle with the radius R_B, the inner circle radius being R_J, the distance between their centers is denoted by e. The inner circle rotates around its center with (clock-wise) rotational speed ω, or we can say, with tangential velocity v_0. It is customary to define the radial clearance $C = R_B - R_J$. As the possible values of e are in the range $e \in \langle 0, C\rangle$ we denote $\varepsilon = e/C$, $\varepsilon \in \langle 0, 1\rangle$ the eccentricity ratio. Hereafter, we say "eccentricity" talking about ε. We can clearly set $R_B = 1$ such that the geometry of our problem is described by two characteristic numbers ε and R_J.

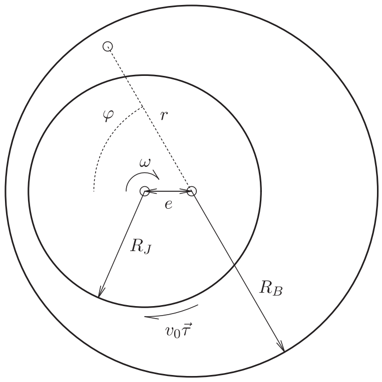


Figure 1: Journal bearing geometry

In practice, the journal is not fixed at all but flows in the lubricant, driven by the applied load on one hand, and by the forces caused by the lubricant on the other hand. Therefore, in the time dependent case the geometry would not be fixed, the journal axis would observe some non-trivial trajectory in the neighborhood of the bearing axis. The simulation would then look somehow as follows: we could set all fluid parameters, the radii of both the bearing and the journal cylinders, prescribe the speed of rotation and the load applied on the journal, and then we could study the trajectory of journal axis in time. Such an approach could be seen e.g. in [3] with many important outcomes concerning the operational regime. One of these observations is that in some cases the motion of the journal axis can cease and can become stable in some "equilibrium" position. Naturally, the position depends on the applied load. In the *steady-state approach*, which we present here, the position of the journal is prescribed and from the solution of lubricant motion we afterwards compute the force applied to the journal by the fluid. By this procedure we obtain the reaction force depending on the eccentricity of cylinders, without performing complex and more time consuming time-dependent simulations. Thus we can effectively study the influence of both geometrical and fluid parameters on the resulting operational regime[2].

2. Governing equations

We consider the lubricant to be a homogeneous incompressible fluid. We do not consider any cavitation in the model, treating only full film of lubricant. The circumstances and effects of cavitation can be found e.g. in [3]. The motion is described by the equations expressing the balance of mass (recall that ρ is a constant)

$$\operatorname{div} \boldsymbol{v} = 0 \quad \text{in } \Omega \tag{1}$$

(we denote Ω the domain of the flow) and the balance of momentum

$$\rho \frac{\partial \boldsymbol{v}}{\partial t} + \rho \sum_{i=1}^{2} v_i \frac{\partial \boldsymbol{v}}{\partial x_i} = \operatorname{div} \boldsymbol{T} + \rho \boldsymbol{b} \quad \text{in } \Omega. \tag{2}$$

As we have decided to study the steady-state problem, the balance of momentum takes the form

$$\rho \sum_{i=1}^{2} v_i \frac{\partial \boldsymbol{v}}{\partial x_i} = \operatorname{div} \boldsymbol{T} + \rho \boldsymbol{b} \quad \text{in } \Omega. \tag{3}$$

For simplicity, we assume that $\rho = 1$ in all what follows. We complete the system (1) and (3) by the Dirichlet boundary condition

$$\boldsymbol{v} = \boldsymbol{v}_0 \quad \text{on } \partial\Omega, \tag{4}$$

such that the lubricant particles on the boundary are supposed to follow the motion of the rigid walls. In the geometry of journal bearing we suppose the outer circle (the bearing wall) to be fixed and the inner circle (the journal) to rotate along its own axis. This means that we prescribe

$$\boldsymbol{v}_0 = \boldsymbol{0} \quad \text{on } \Gamma_{\mathrm{O}} \subset \partial\Omega \text{ (the outer circle)} \tag{5}$$

$$\boldsymbol{v}_0 = v_0 \boldsymbol{\tau} \quad \text{on } \Gamma_{\mathrm{I}} \subset \partial\Omega \text{ (the inner circle),} \tag{6}$$

where v_0 is given and $\boldsymbol{\tau} = \boldsymbol{\tau}(\boldsymbol{x})$ is the (clock-wise) unit tangential vector to the inner circle Γ_{I}.

The crucial step now is to set the model of Cauchy stress tensor $\boldsymbol{T}$.

A fluid is called Newtonian if the dependence of the stress tensor on the spatial variation of velocity is linear. This model was introduced by Stokes[3] in 1844 [1] and already Stokes remarked that the model may be applicable to fluid flows at normal conditions. For instance, while the dependence of the viscosity on the pressure does not show up in certain common flows, it can have a significant effect when the pressure becomes very high.

As the lubricant in journal bearing is forced through a very narrow region, of order of micrometers, the pressure can become so high that the fluid obtains a "glassy" state. Moreover, since the shear-rate becomes also high, the viscosity of the lubricant does not suffice to be considered constant with respect to the shear-rate. We thus consider the Cauchy stress tensor to be of the form

$$\boldsymbol{T} = -p\boldsymbol{I} + \rho\nu(p, |\boldsymbol{D}|^2)\boldsymbol{D}, \tag{7}$$

where

$$|\boldsymbol{D}|^2 = \operatorname{tr} \boldsymbol{D}^2, \qquad \boldsymbol{D} = \frac{1}{2}\left(\nabla \boldsymbol{v} + (\nabla \boldsymbol{v})^T\right).$$

[2] Also, knowing the dependence of the reaction force on the eccentricity, we can proceed to "quasi-stationary approach" solving the system of ODEs for the journal axis trajectory, assuming that at each time step the flow of the lubricant is "steady". On the other hand, one of the disadvantages of this approach is that knowing the position of the journal and the corresponding reaction force, we still do not know anything about the stability of such a configuration. Anyway, this questions are out of the scope of this work.

[3] the model was earlier introduced also by Navier and Poisson

For details concerning the derivation of this class of constitutive models see e.g. [5, 6, 7, 8, 10]

As we introduce the new variable p into the equations, the pressure, we have to add the condition[4]

$$\frac{1}{|\Omega|}\int_\Omega p \mathrm{d}x = p_0, \tag{8}$$

where p_0 is given constant. We note that as soon as the pressure figures in the viscosity formula the value of p_0 is no more dismissible, but it can affect the behavior of the solution, in the contrary to the case of Navier-Stokes equations (see [9]).

The dependence of the viscosity on the pressure has been studied for quite a long time. For instance in the magisterial treatise of Bridgman [2] there is a discussion of the studies up to 1931. The dependence of the viscosity on the pressure is mostly considered to be exponential, the simple form

$$\nu(p) = \exp(\alpha p) \tag{9}$$

is often used. As a representative of models where the viscosity depends only on the shear-rate we mention the (shear-thinning) power-law model

$$\nu(\boldsymbol{D}) = \nu_0 |\boldsymbol{D}|^{p-2}, \quad p \in (1,2).$$

Here we test a different viscosity formula, following the recent positive results in the existence theory.

Although the fluid models with the pressure dependent viscosities are studied and used at least from the first third of the last century, mathematical results concerning the existence of solutions are rare. Recently, the global-in-time existence of solutions for a class of fluids with the viscosity depending not only on the pressure but also on the shear rate was established – see [6, 5, 7]. These results, established under the assumption that the flow is spatially periodic, was achieved also for homogeneous Dirichlet boundary condition in [8]. This was generalized to the case of non-homogeneous Dirichlet condition in the paper [10], provided that only a tangential component of the velocity is nonzero on the boundary, i.e. under the assumption that

$$\boldsymbol{v} \cdot \boldsymbol{n} = 0 \quad \text{on } \partial\Omega \tag{10}$$

($\boldsymbol{n}$ means a normal vector to $\partial\Omega$). Note that the boundary condition (4)-(6) given for the journal bearing fulfills (10). These results strongly use the fact, that the viscosity does not depend only on the pressure, but also (in a suitable way) on the shear-rate. For detailed assumptions made on this subclass of (7) see [5, 6, 7, 8, 9, 10]. Let us only mention, that any exponential model of the form

$$\nu(p, |\boldsymbol{D}|^2) = \nu_{\boldsymbol{D}}(|\boldsymbol{D}|^2)\exp(\alpha p) \tag{11}$$

or similar does not meet the assumptions and, up to our knowledge, the existence of a solution for such a case is not clear.

Here, we examine the model, also introduced and numerically studied in [7], of the following form:

$$\nu(p, |\boldsymbol{D}|^2) = \nu_0 \left(A + (\beta + \exp(\alpha p))^{-q} + |\boldsymbol{D}|^2\right)^{\frac{r-2}{2}}. \tag{12}$$

This model (under some additional assumptions on the positive constants ν_0, A, α, β, q and[5] on $r \in (\frac{3}{2}, 2)$) meets the assumptions given in [10] (see [9] for more details) and thus the theoretical results given in [10] can be applied. Let us point out the properties of (12) briefly. The viscosity $\nu(\cdot,\cdot)$ is decreasing function of $|\boldsymbol{D}|^2$ and increasing function of p (as soon as $r < 2$). For $|\boldsymbol{D}|$ great enough the remaining terms are bounded by $A + \beta^{-q}$ and the shear dependence becomes dominant. Asymptotically, (12) behaves like the power-law model:

$$\nu(p, |\boldsymbol{D}|^2) \sim \nu_0 |\boldsymbol{D}|^{r-2}, \quad \text{as } |\boldsymbol{D}| \to \infty, \ p \text{ arbitrary}.$$

[4] or some similar condition on the level of the pressure field

[5] Our theoretical results give the existence only for $r \in (\frac{3}{2}, 2)$. However, we provided the simulations with r being within the range $r \in (1, 2)$ and, in our geometry, we did not observed any significant change of (numerical) behavior near the value $r = \frac{3}{2}$.

We see that in some feasible range of pressure (such that β can be neglected, but in the same moment $\exp(-q\alpha p) >> A + |\boldsymbol{D}|^2$)

$$\nu(p, |\boldsymbol{D}|^2) \sim \nu_0 \exp(\alpha p)^{q\frac{2-r}{2}}, \quad \text{for } A, |\boldsymbol{D}|^2 << 1 \text{ and } p \text{ in some feasible range},$$

such that the model is in this sense similar to the relation (9). Unfortunately, for the pressure being large, asymptotically

$$\nu(p, |\boldsymbol{D}|^2) \sim \nu_0 (A + |\boldsymbol{D}|^2)^{\frac{r-2}{2}}, \quad \text{as } p \to \infty, |\boldsymbol{D}| \text{ fixed},$$

the viscosity is bounded, its supremum being $\nu_0 A^{\frac{r-2}{2}}$. This property necessarily follows from the method used in [10] in order to prove the existence of solutions.

In order to keep the similarity of our model with (9), we set

$$q := \frac{2}{2-r} \quad , \quad A := \beta := 10^{-5}. \tag{13}$$

3. Non-dimensional form of the equations

In the classical Navier-Stokes equations it is customary to characterize the flow problem by the non-dimensional Reynolds number, defined as

$$\mathrm{Re} = \frac{UV}{\nu},$$

where U and V are the characteristic length and the characteristic velocity, respectively. This is a consequence of the fact, that if we introduce these characteristic quantities into equations, writing them in terms of non-dimensional velocity, pressure and length, the only term that remains in the equations is exactly the Reynolds number. This is no longer true as soon as we consider that the viscosity depends non-trivially on the pressure and/or on the velocity gradient.

Let us consider the classical model with constant viscosity as an approximation of the generalized one, in the case when the pressure and the shear-rate are not too great. Following this idea, it seems to be reasonable to define a quantity

$$\mathrm{Re}^* := \frac{UV}{\hat{\nu}_0}, \quad \text{where } \hat{\nu}_0 := \nu(0,0). \tag{14}$$

The momentum equation (3) then transforms to the non-dimensional form

$$\hat{v}_i \frac{\partial \hat{\boldsymbol{v}}}{\partial \hat{x}_i} + \nabla \hat{p} - \frac{1}{\mathrm{Re}^*} \operatorname{div} \left(\hat{\nu}(\hat{p}, |\hat{\boldsymbol{D}}|^2) \hat{\boldsymbol{D}} \right) = \hat{\boldsymbol{b}}, \tag{15}$$

where we define the modified viscosity form

$$\hat{\nu}(\hat{p}, |\hat{\boldsymbol{D}}|^2) := \frac{1}{\hat{\nu}_0} \nu \left(p, |\boldsymbol{D}|^2 \right) = \frac{1}{\hat{\nu}_0} \nu \left(V^2 \hat{p}, \frac{V^2}{U^2} |\hat{\boldsymbol{D}}|^2 \right). \tag{16}$$

Let us note that $\hat{\nu}(0,0) = 1$.

4. Numerical method

We briefly discuss the numerical method used to obtain the approximations of the solution. We use the software package `featflow`, the finite element method package developed initially to solve the Navier-Stokes equations and modified in order to solve also the Navier-Stokes-like systems with the non-constant viscosity. For the information concerning the basic methods used in the package, concerning the efficiency and the mathematical background, as well as for the software itself, we refer to `www.featflow.de`, the featflow manual [12] and the book by S. Turek [11].

The triangulation of the domain is done via quadrilateral elements. The $\tilde{Q}1/Q0$ Stokes element uses "rotated bilinear" shape functions for the velocity and piecewise constants for the pressure. One of the features of this element choice is that it admits the simple upwind strategies which lead to matrices with better properties, these methods are included in `featflow` and we use it without providing any further description. For details see [11].

For the definition of the discrete weak solution to our problem see [9]. The discrete formulation leads to the system of nonlinear algebraic equations, that are solved via the adaptive fixed point defect correction method (solving an Oseen-like subproblem at each iteration). Linear problems resulting in each step are solved by a multi-grid method, where Vanka-like block-Gauß-Seidel scheme is used both as a smoother and a solver. For all details, documentation and further analysis we refer to [11, 12].

Since we use the formulation including the symmetric part of the velocity gradient $\boldsymbol{D}$, this approach in itself is unstable due to the failure to satisfy a discrete Korn's inequality. The stabilization technique is thus included in the code, see [13] for reference.

5. Numerical results

The first component of this section are the results of the classical Navier-Stokes model applied to the journal bearing geometry. The main aim is to show the influence of the varying eccentricity of the journal and the behavior of the Navier-Stokes model with various Reynolds numbers. In all simulations we set the velocity prescribed on the inner circle to be 1, i.e. to be the characteristic velocity of the real problem. Similarly, we set 1 the radius of the outer circle. The radius of the inner circle we set to be 0.8, which gives us the possible range of absolute eccentricity $\varepsilon \in (0,1) \approx e \in (0, 0.2)$.

The resulting (non-dimensional) pressure $\hat{p}$ distributions (we set $\hat{p}_0 = 0$) for the Reynolds numbers 1, 100 and 1000 and for the eccentricities 0.3 and 0.8 are shown in figure 2. In figure 3 there are shown the distributions of $|\hat{\boldsymbol{D}}|$ and figure 4 shows the streamlines of the resulting flow.

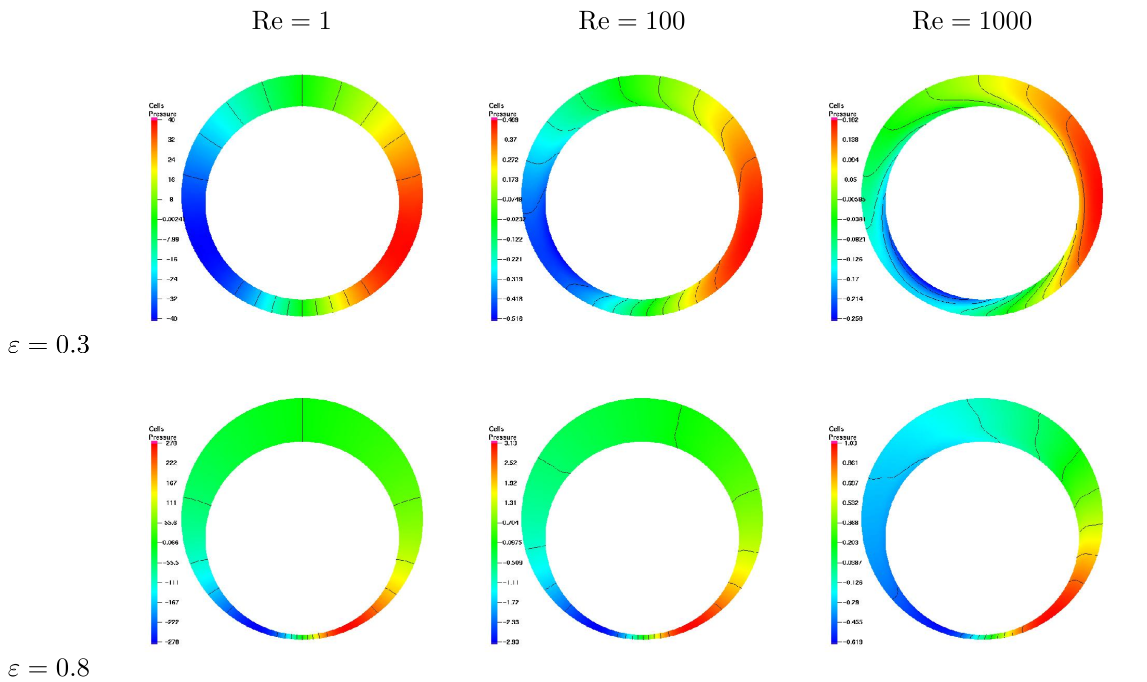


Figure 2: The pressure $\hat{p}$ distribution for the Navier-Stokes model

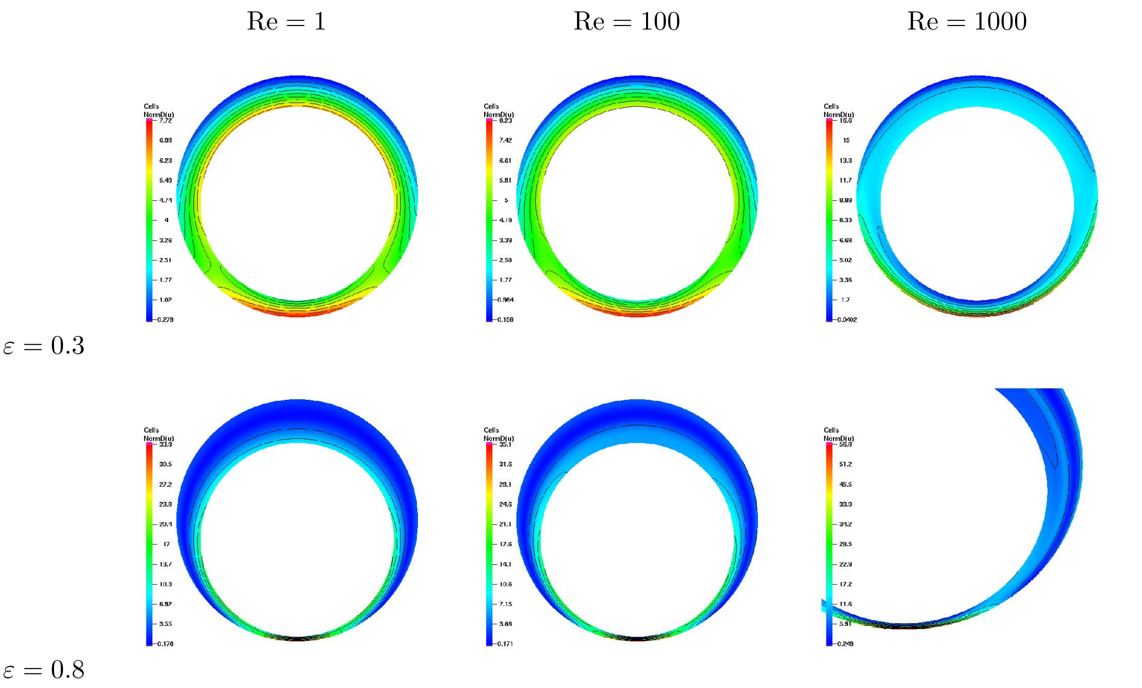


Figure 3: $|\hat{\boldsymbol{D}}|$ distribution for the Navier-Stokes model

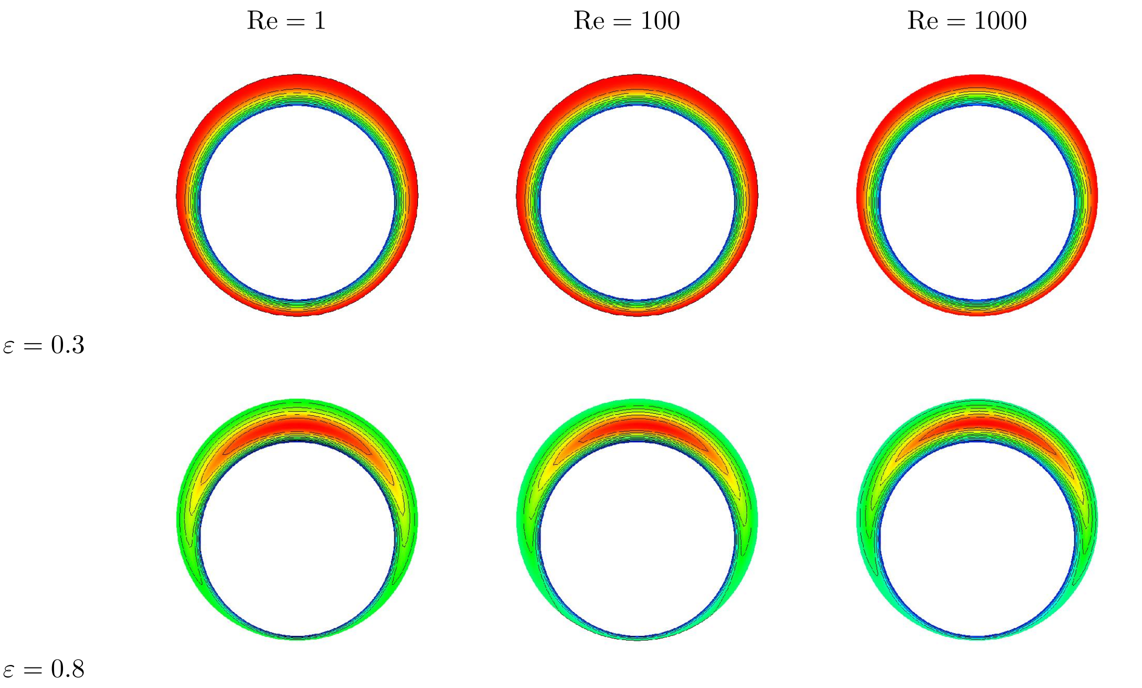


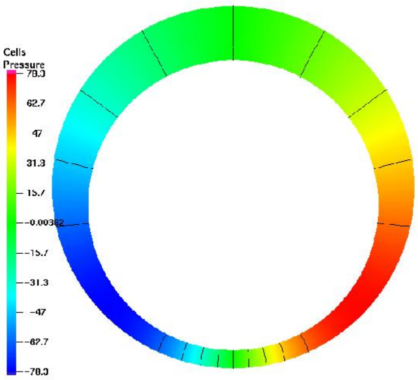
Figure 4: The stream-lines for the Navier-Stokes model

In the second part we show the differences that occur when we introduce the generalized model, with the viscosity of the form (12). We set $r = 1.5$ as the lower border of the range within we proved the existence, and in order to keep (13) we set $q = 4$. As we have already mentioned, we set $A := \beta := 10^{-5}$. Motivated by [3] we set $\alpha := 10^{-8}$.

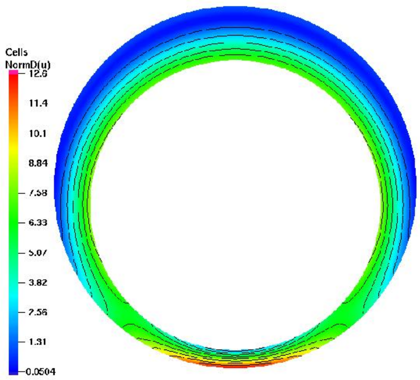
Let us look to the previous simulations. We can see, observing for instance the case $\varepsilon = 0.5$, $\mathrm{Re} = 1$ in figure 5, that on the majority part of the flow domain $|\hat{\boldsymbol{D}}|^2 \sim 30$. On the other side, the (non-dimensional) pressure results somewhere in the range $\hat{p} \sim -100.. + 100$. Since we would like to have a positive pressure values (realizing that in the pressure-dependent viscosity case this becomes important, in contrast to the Navier-Stokes case) we set the pressure mean value $\hat{p}_0 = 100$ such that we can expect $\hat{p} \sim 0..200$ from the Navier-Stokes case. If we would set $U = V = 1$ in the non-dimensional transformation (16), we would obtain $(\beta + \exp(\alpha\hat{p}))^{-q} \sim$ from $1 - 10^{-5}$ to 1. Note that setting the Reynolds number higher than one we obtain even smaller range of the pressure field (in the Navier-Stokes case). Naturally, we wouldn't obtain any visible dependence of the viscosity on the pressure setting the parameters in this manner.

We recall the non-dimensional transformation (16), which we employ in order to balance the pressure- and the shear- dependence in (12) in such a way that we can demonstrate the abilities of the model. We keep $\mathrm{Re}^* = 1$ in what follows - from now, the influence of the convective term is not in the center of our interest.

First, we set the characteristic velocity V in such a way that $(\beta + \exp(\alpha V^2\hat{p}))^{-q} \sim 0.5$ for $p \sim 200$. We thus set $V = 300$. We notice that for $p \sim 100$, that is for the prescribed mean value of the pressure, we obtain $(\beta + \exp(\alpha p))^{-q} \sim 0.7$. Therefore, as a second step, we set the characteristic length U such that for $|\hat{\boldsymbol{D}}|^2 \sim 30$ we obtain $|\boldsymbol{D}|^2 = \frac{V^2}{U^2}|\hat{\boldsymbol{D}}|^2 \sim 0.7$ We thus set $U = 2000$. As we have decided to keep $\mathrm{Re}^* = 1$ for now, we must set ν_0 such that $\hat{\nu}_0 = \nu(0,0) := UV = 6 \times 10^5$. The last choice is, of course, a very unrealistic one. This should be understood as a numerical experiment, preliminary to the further studies of the real-case parameters and geometry, which are considered as the next step.



values of the pressure $\hat{p}$

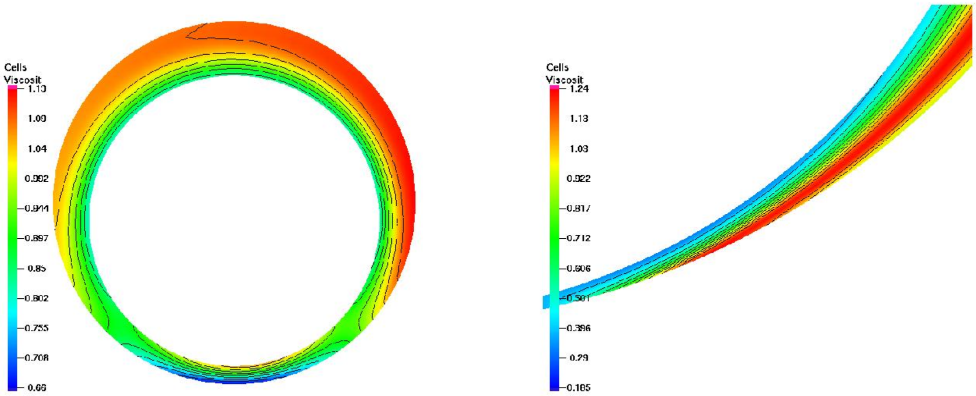


values of $|\hat{\boldsymbol{D}}|$

Figure 5: Some Navier-Stokes results for $\varepsilon = 0.5$

In figure 6 we show the viscosity field for the case $\varepsilon = 0.5$. In table 1 we present the comparison of the following quantities for the Navier-Stokes and for the model (12): we show the minimum and maximum values of the pressure (we shifted the pressure mean value to 100 in the Navier-Stokes case), of the shear $|\hat{\boldsymbol{D}}|$ and of the viscosity, then we show the (non-dimensional) force magnitude and its direction.

In table and graph 2 we show the minimum and maximum viscosities for several eccentricities for $\mathrm{Re}^* = 1$. In tables 3, 4 and 5 we present the maximum pressure and $|\hat{\boldsymbol{D}}|$ values, the force magnitude and its direction,



$\varepsilon = 0.5$ $\varepsilon = 0.95$

Figure 6: The viscosity field for the model (12), $\mathrm{Re}^* = 1$

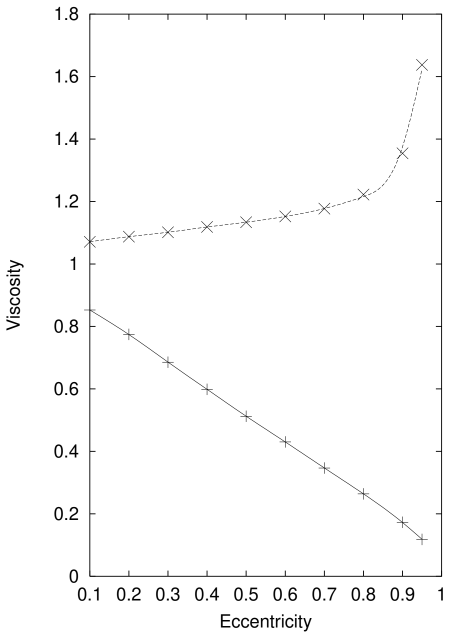
	$\hat{p}_{\min}$	$\hat{p}_{\max}$	$\lvert\hat{\boldsymbol{D}}\rvert_{\min}$	$\lvert\hat{\boldsymbol{D}}\rvert_{\max}$	$\hat{\nu}_{\min}$	$\hat{\nu}_{\max}$	force mag.	force dir.
N.-S.	22	178	0.05	12.6	1	1	172	0.4998
model (12)	41	159	0.002	612	0.51	1.13	137	0.5011

Table 1: A comparison between N.-S. and the model (12), $\mathrm{Re}^* = 1$, $\varepsilon = 0.5$

compared with the Navier-Stokes case for several eccentricities.

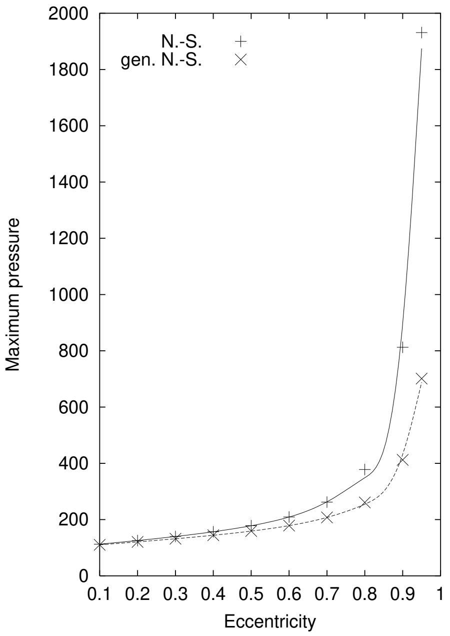
6. Conclusion

We presented one sample form of the viscosity which fulfills the conditions of the recent existence result, and we shown that it is indeed able both of significant shear thinning and pressure thickening effects, so important in the context of the journal bearings. The eccentricity influence was systematically studied in order to compare the behavior of our generalized model with the Navier-Stokes fluid in one selected example. The main aim was not to give any engineering prediction or quantitative results but to show the extended capabilities of the generalized model same as the need to determine and set the additional parameters occurring in the model.



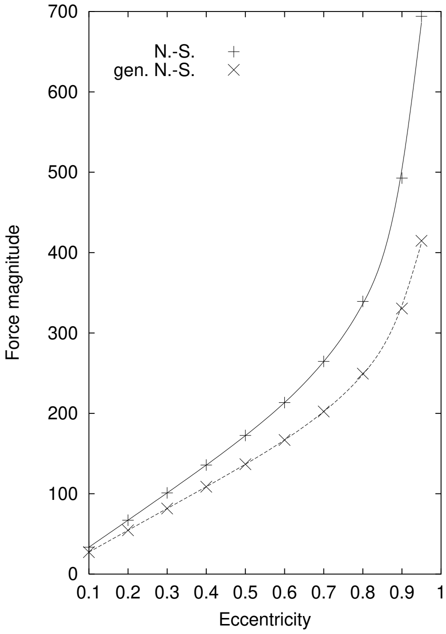
	$\hat{\nu}_{\min}$	$\hat{\nu}_{\max}$
$\varepsilon = 0.1$	0.85	1.07
0.2	0.77	1.09
0.3	0.69	1.10
0.4	0.60	1.12
0.5	0.51	1.13
0.6	0.43	1.15
0.7	0.35	1.18
0.8	0.26	1.22
0.9	0.17	1.35
0.95	0.12	1.64

Table 2: The minimum and maximum viscosities, $\mathrm{Re}^* = 1$



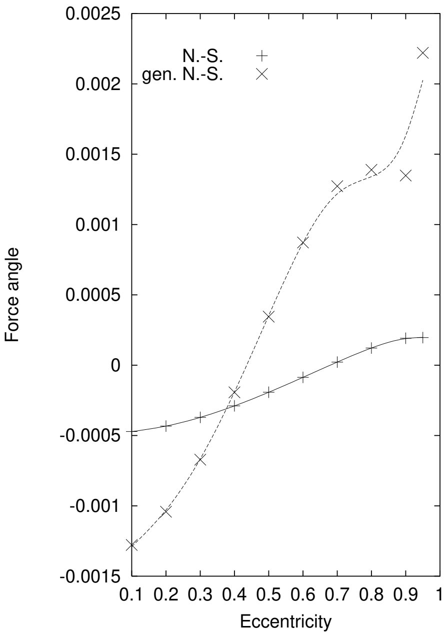
	N.-S.	gen. N.-S.
$\varepsilon = 0.1$	113	110
0.2	126	121
0.3	140	132
0.4	157	144
0.5	178	159
0.6	209	179
0.7	262	208
0.8	378	261
0.9	813	412
0.95	1931	701

Table 3: Maximum pressure $\hat{p}$ values, N.-S. and the model (12), $\mathrm{Re}^* = 1$



	N.-S.	gen. N.-S.
$\varepsilon = 0.1$	34	27
0.2	67	55
0.3	101	82
0.4	136	109
0.5	172	137
0.6	213	167
0.7	265	202
0.8	339	249
0.9	493	330
0.95	694	415

Table 4: Force magnitude, comparison between N.-S. and the model (12), $\mathrm{Re}^* = 1$



	N.-S.	gen. N.-S.
$\varepsilon = 0.1$	-0.00047	-0.0013
0.2	-0.00043	-0.0010
0.3	-0.00037	-0.0007
0.4	-0.00029	-0.0002
0.5	-0.00019	0.0003
0.6	-0.00009	0.0009
0.7	0.00002	0.0013
0.8	0.00012	0.0014
0.9	0.00019	0.0014
0.95	0.00020	0.0022

Table 5: Force direction, comparison between N.-S. and the model (12), $\mathrm{Re}^* = 1$

References

[1] G. G. Stokes, "On the theories of the internal friction of fluids in motion, and of the equilibrium and motion of elastic solids," *Trans. Cambridge Phil. Soc.*, vol. 8, pp. 287–305, 1845.

[2] P. W. Bridgman, "The physics of high pressure," *the MacMillan Company, New York*, 1931.

[3] D. Rh. Gwynllyw, A. R. Davies, and T. N. Phillips, "On the effects of a piezoviscous lubricant on the dynamics of a journal bearing," *Journal of Rheology*, vol. 40, pp. 1239–1266, Nov. 1996.

[4] A. Z. Szeri, "Fluid film lubrication: theory and design," *Cambridge University Press*, 1998.

[5] J. Málek, J. Nečas, and K. R. Rajagopal, "Global existence of solutions for flows of fluids with pressure and shear dependent viscosities," *Applied Mathematics Letters*, vol. 15, pp. 961–967, Nov. 2002.

[6] J. Málek, J. Nečas, and K. R. Rajagopal, "Global analysis of the flows of fluids with pressure-dependent viscosities," *Archive for rational mechanics and analysis*, vol. 165, pp. 243–269, Dec. 2002.

[7] J. Hron, J. Málek, J. Nečas, and K. R. Rajagopal, "Numerical simulations and global existence of solutions of two dimensional flows of fluids with pressure and shear dependent viscosities," *Mathematics and Computers in Simulation*, vol. 61, pp. 297–315, 30 Jan. 2003.

[8] M. Franta, J. Málek, and K. R. Rajagopal, "On steady flows of fluids with pressure- and shear- dependent viscosities," *Proceedings of the Royal Society A - Mathematical Physical and Engineering Sciences*, vol. 461, pp. 651–670, 8 Mar. 2005.

[9] M. Lanzendörfer, "Numerical Simulations of the Flow in the Journal Bearing," *MS-thesis on Charles University in Prague, Faculty of Mathematics and Physics*, 2003.

[10] M. Lanzendörfer, "On non-homogeneous Dirichlet boundary conditions for planar steady flows of an incompressible fluid with the viscosity depending on the pressure and the shear rate," *WDS'05 Proceedings of Contributed Papers: Part III - Physics (ed. J. Šafránková), Prague, Matfyzpress*, pp. 619-624, 2004.

[11] S. Turek, "Efficient solvers for incompressible flow problems, An Algorithmic and Computational Approach," *Springer-Verlag Berlin Heidelberg*, 1999.

[12] S. Turek, Chr. Becker, "FEATFLOW, Finite element software for the incompressible Navier-Stokes equations, User Manual (Release 1.1)," *to be found on* `www.featflow.de`.

[13] S. Turek, A. Ouazzi, R. Schmachtel, "Multigrid methods for stabilized nonconforming finite elements for incompressible flow involving the deformation tensor formulation," *Journal of Numerical Mathematics*, vol. 10, no. 3, pp. 235-248, 2002.

Data Integration in VirGIS and in the Semantic Web

Post-Graduate Student:
ING. ZDEŇKA LINKOVÁ
Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2
182 07 Prague 8
linkova@cs.cas.cz

Supervisor:
ING. JÚLIUS ŠTULLER, CSC.
Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2
182 07 Prague 8
stuller@cs.cas.cz

Field of Study:
Mathematical engineering
Classification: 39-10-9

This work was supported by the project 1ET100300419 of the Program Information Society (of the Thematic Program II of the National Research Program of the Czech Republic) "Intelligent Models, Algorithms, Methods and Tools for the Semantic Web Realization", by the project of Czech-French Cooperation Barrande 2004-003-1, 2: "Integration de données sur le Web - applications aux Systémes d'Information Géographique (2003-2005), and by the Institutional Research Plan AV0Z10300504 "Computer Science for the Information Society: Models, Algorithms, Applications".

Abstract

Integration has been an acknowledged data processing problem for a long time. However, there is no universal tool for general data integration. Because various data descriptions, data heterogeneity, and machine unreadability, it is not easy way. Improvement in this situation could bring the Semantic Web. Its idea is based on machine understandable web data, which bring us an opportunity of better automated processing. The Semantic Web is still a future vision, but there are already some features we can use. The paper describes how is integration solved in mediation integration system VirGIS and discusses use of nowadays Semantic Web features to improve it. According to the proposed changes, a new ontology that covers data used in VirGIS is presented.

1. Introduction

Today's world is a world of information. Expansion of World Wide Web has brought better accessibility to information sources. However, in the same time, the big amount of different formats, data heterogeneity, and machine unreadability of this data have caused many problems. One of them is a problem of integration. To integrate data could mean to provide one global view over several data sources and let them be processed as one source. To integrate data means to provide one global view over different data sources [1]. This view can be either materialized, or virtual. An important thing is to combine data in meaningful way and let them be accessible as one whole. There are two main problems resulting from the data integration. The first is the data modeling (how to integrate different source schemas); the second is their querying (how to answer to the queries posed on the global schema). The integration process is not easy. Yet, there is no universal tool or method that could be used every time when needed. Nevertheless, there are some partial solutions in many research areas.

As mentioned above, data features make automated processing difficult. Exactly from this base rises the idea of the Semantic Web [2]. It considers data to go along with their meanings. An addition of semantics would make data machine readable and understandable. The automation could be easier. This proposal is for general web data, so it offers to use it also for specialized kind of data.

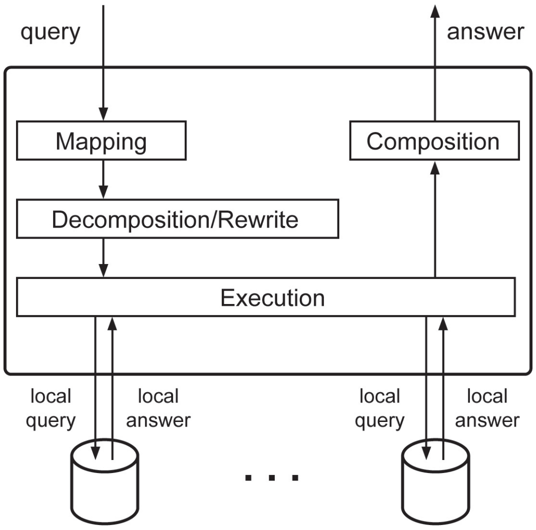


Figure 1: VirGIS System

Integration has been solved also in the area where GIS (Geographic Information Sources) [3] are used. Among these solutions, there is also VirGIS, an integration system that work with satellite images.

2. VirGIS System

VirGIS [4] is a mediation platform that provides an virtual integrated view of geographic data. In general, the main idea in a global virtual view use is a system of components called mediators. Mediators provide an interface of the local data sources. There are also other special components - wrappers, which play the roles of connectors between local source backgrounds and the global one. The principle of integration is to create a nonmaterialized view in each mediator. These views are then used in the query evaluation. Essential are mapping rules that express the correspondence between the global schema and the data source ones. The problem of answering queries is another point of the mediation integration - a user poses a query in terms of a mediated schema, and the data integration system needs to reformulate the query to refer to the data sources.

VirGIS accesses GIS data sources via Web Feature Service (WFS) server and uses WFS interfaces to perform communications with sources. WFSs play the role of wrappers in the mediation system. VirGIS uses GML as an internal format to represent and manipulate geographic information. GML is a geographic XML-based language; therefore GQuery, a geographic XQuery-based language, is used for querying. The integration system has only one mediator called GIS Mediator. It is composed of a Mapping module, a Decomposition/Rewrite module, an Execution module and Composition module.

The Mapping module uses integrated schema information in order to express user queries in terms of local source schemas. Each mapping rule expresses a correspondence between global schema features and local ones. For the global schema definition, a Local As View (LAV) approach is applied. This approach consists in defining the local sources as a set of views made on the global schema. In current version of VirGIS, there are used simple mapping rules that allow the specification of one-to-one schema transformations under some constraints: aggregations and one-to-many mappings are not considered. The Decomposition/Rewrite module exploits information about source feature types and source capabilities to generate an execution plan. A global GQuery expression is used as a container for collecting and integrating results coming from local data sources. The Execution module processes sub-queries contained in the execution plan and sends them to the appropriate source's WFS. The Composition module treats the final answer to delete duplicities and produces a GML document, which is returned to the user.

3. Use of Semantic Web features in mediation integration system

The Semantic Web is intended as an extension of today's World Wide Web. It should consist of machine readable, understandable and meaningfully processable data. The basis is addition of data semantics - there will be stored data meaning description together with data themselves. The Semantic Web idea belongs still to the future; however, there have been made already some features. It is based on standards, which are defined by W3C (WWW Consortium) [5]. The Semantic Web could improve or make easier to automate some operations. Hopefully it could bring something more also in data integration process. There are some areas, which could benefit by better automatization; for example addition of new sources, mapping rules generation and schema evolving.

3.1. Data sources

An important requirement of machine processable information is data structuring. On the web nowadays, the language XML (eXtensible Markup Language) [6] is used for making web document structure. But only XML is not enough to describe data. The technique to specify the meaning of information is RDF (Resource Description Framework) [7]. It is basic tool of web sources metadata addition. RDF data model gives an abstract conceptual framework for metadata definition and usage. It uses XML syntax (RDF/XML) for encoding. Additionally, there is also an extension of RDF called RDF Schema [8] that is useful for class definition and class hierarchy description. Instruments for definition of terms used either in data or in metadata are ontologies. In the context of web technologies, ontology is a file or a document that contain formal definitions of terms and term relations. The Semantic Web technique for definition of ontologies is the OWL (Ontology Web Language) [9] language.

In the VirGIS integration system, an XML-based language is used for data representation. If the integration is XML-based, why not bring more and, instead of simple XML, use RDF, which has bigger expressive power. So in the proposed integration system, the RDF is intended to represent information. Also XML document primarily not intended for RDF applications could be described using RDF. By observing several guidelines when designing the schema, [10] proposed how to make an XML "RDF-friendly". For already existing documents, there is possibility to make some XML-RDF bridge. Of course, it has not to be always simple way.

As with data, the XML and RDF worlds use different formalism for expressing schema. The Semantic Web currently uses languages such as RDFS and OWL. So in the proposed integration system, OWL is used to publish sets of terms (called ontologies). Of course a source can use some richer ontology (richer than the source need as the schema). In this case, the source schema can be seen as a view of the ontology.

3.2. Querying

According to data description change, a change in querying is needed. Since RDF is defined using an XML syntax, it might appear on the first sight, that a query language and system for XML would also be applicable to RDF. This is, however, not the case, since XML encodes the structure of data and documents whereas the RDF data model is more abstract. The relations or predicates of the RDF data model can be user defined and are not restricted to child/parent or attribute relations. A query language based on XML element hierarchies and attribute names will not easily cope with the aggregation of data from multiple RDF/XML files. Also, the fact that RDF introduces several alternative ways to encode the same data model in XML means that syntax-oriented query languages will be unable to query RDF data effectively. Having motivated the need of an RDF query language, there was developed some query languages. A standardized query language for RDF data is called SPARQL [11].

3.3. Mapping and query rewriting

Essential task for the integration system are mapping rules and query rewriting, too. Closely related with it is also new sources addition and how (or whether) it could be done automatically. Mapping rules in VirGIS are expressed utilizing XML. However, the idea about the improvement of the integration system is to be able apply existing mapping rules, knowledge about already integrated sources, and knowledge about the new one to generate (automatically as much as possible) appropriate new mapping rules. Doing this, taking advantage of an inference mechanism tool would be practicable. But it requires machine processable data. Similarly to data sources, there is an idea to use RDF/XML instead of this pure XML. Nevertheless, even RDFS has no construct for terms or classes equivalency expression. There must be used some additional capabilities.

A possibility is own development to enrich RDF(S). Another possibility is to work with OWL, which is standard extension of RDFS. Using OWL provides at least two approaches. The first way is definition of mapping rules as a special class. The second way is to present mapping between schemas and concepts of sources by usage of OWL construct in order to express equivalency of some parts of different sources ontologies. The same situation is also in field of query rewriting. It needs further study. Of course, there some existing algorithms that could be used. Or, this could be improved, according to chosen technique of mapping rules definition, cleverness of particular local sources query mechanism, and potentialities of an accessible tool that implements SPARQL.

4. Building ontology for VirGIS system

The first step towards a Semantic Web-based version of integration system VirGIS was VirGIS ontology development. This task was joint work with Radim Nedbal[1]. Our aim was to build an ontology for a given data domain; it had cover at least data provided by VirGIS.

The term “ontology” has been used in many ways and across different communities. A popular definition of the term ontology in computer science is: an ontology is a formal, explicit specification of a conceptualization. A conceptualization refers to an abstract model of some phenomenon in the world. However, a conceptualization is never universally valid. Ontologies have been set out to overcome the problem of implicit and hidden knowledge by making the conceptualization explicit. An ontology may take a variety of forms, but it will necessarily include a vocabulary of terms and some specification of their meaning.

There are many tools and languages [12] that can be employed as means for ontology development. Among available ontology languages, Web Ontology Language (OWL) was chosen. OWL is proposed to be an ontology language for the Semantic Web. OWL, a XML based language, has more facilities for expressing meaning and semantics than XML, RDF, and RDF Schema, and thus OWL goes beyond these languages in its ability to represent machine interpretable content on the Web. OWL adds more vocabulary for describing properties and classes. A large number of organizations have been exploring the use of OWL, with many tools currently available.

As an ontology design tool, Protégé System [13] was used. Protégé is an integrated software tool used to develop ontologies and knowledge-based systems. Protégé has been developed by the Stanford Medical Informatics (SMI) at Stanford University.

4.1. VirGIS data

VirGIS is implemented as an integration system of satellite images. Figure 2 illustrates local and global sources of VirGIS. As local sources are used subsets of schemas drawn from SPOT and IKONOS catalogues and QUICK_LOOK database.

SPOT and IKONOS catalogues provide information about satellites; QUICK_LOOK refers to a sample of small images that give an overview of satellite images supplied in the catalogue. The role of the global source is played by the VIRGIS mediated schema. The VIRGIS schema contains just one entity VIRGIS with following attributes:

- string *id* (a common id for the different region photographed)
- string *name* (the name of the satellite that takes the photo)

[1] nedbal@cs.cas.cz

SPOT	
Attribute	Type
date_	Date
sun_elev	numeric
satellite	string
sat_id	numeric
key	string
the_geom.	Polygon

IKONOS	
Attribute	Type
date_acqui	Date
sun_el	numeric
satellite	string
sat_id	numeric
key	string
the_geom	Polygon

VIRGIS	
id	string
name	string
satid	string
date	Date
sun_elevation	numeric
url	string
geom	Polygon

QUICK LOOK	
Attribute	Type
key	string
filename	string

Figure 2: Local and global satellite schemas

- string *satid* (the id for the satellite)
- date *date* (the date when the photo was taken)
- numeric *sun_elevation* (the sun elevation when photo was taken)
- string *url* (the url where the real photo is saved)
- polygon *geom* (the geometry of the region photographed)

According to this schema description, the aim was a development of an ontology satisfying the VirGIS data semantics. It had to cover not only the global schema, but also the local ones and relationships among them.

4.2. The VirGIS ontology

The aim was a description of satellite image knowledge in a VirGIS ontology. In ontology re-use, we can consider only some general spatial ontology for basic geometric features. The VirGIS data area itself is not covered with any existing GIS ontology. A new ontology for this purpose is needed.

The proposed VirGIS specified ontology comes out of the data model described above. The main domain concepts and their relationships are depicted in Figure 3 by means of ISA tree.

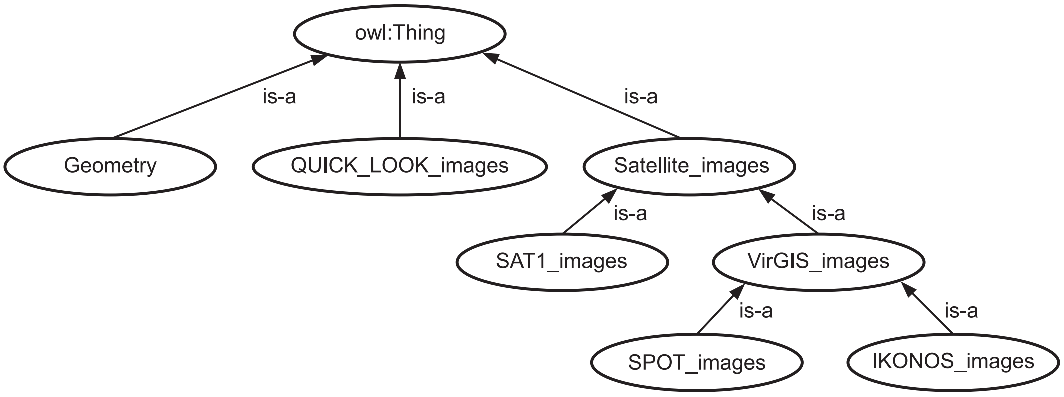


Figure 3: ISA diagram of the model

Observe that each node corresponds to one concept. `IKONOS_images` and `SPOT_images` refer to local sources; `VirGIS_images` refers to the global mediated source. The fact that every image contained in IKONOS or SPOT database is also contained in VirGIS induces the corresponding concepts relationship that can be understood as set inclusions:

$$\begin{aligned} IKONOS\_images &\subseteq VirGIS\_images, \\ SPOT\_images &\subseteq VirGIS\_images, \end{aligned} \tag{1}$$

Analogical relationship applies to `VirGIS_images` and `Satelite_images` concepts. Observe that there is an additional class `SAT1_images` in the model. It contains satellite images not integrated in VirGIS_images. Finally, an inherent feature of the OWL data model is the unique superclass `THING` being the superclass of all other classes.

In OWL, a `owl:Class` construct is used for concept indication and `rdfs:subClassOf` construct for expressing the concept relationships corresponding to set inclusion relations:

Example 6 *The OWL expression of the relationship of* `SPOT` *and* `VirGIS` *classes*

```
<owl:Class rdf:ID="SPOT_images">
   <rdfs:subClassOf rdf:resource="#VirGIS_images" />
</owl:Class>
```

The `rdfs:subClassOf` construct expresses inclusion relationship on both set and conceptual level. Therefore, the above OWL code example implies SPOT_images being conceptually more specific than VirGIS_images.

In OWL, classes are also characterized by means of properties, i.e. attributes of corresponding concepts. Properties definitions are to represent the semantic relationships of the corresponding concepts and their attributes.

Observe that SPOT and IKONOS use semantically equivalent attributes without any common name convention. In addition, VirGIS introduces its own identifiers for respective attributes. `date_` (SPOT), `date_acqui` (IKONOS) and `date` (VirGIS) represent semantically equivalent attributes for instance. This is solved with mapping of mediation integration in VirGIS. However, it can naturally be expressed on the semantic level, by means of OWL.

With regard to the above discussion and considering the inclusion 1, it follows:

$$(\forall image \in SPOT\_images)(date\_(image, DD/MM/YY) \rightarrow date(image, DD/MM/YY)),$$

which defines the semantic relationship of the binary predicates `date_` and `date`. The relationships between other predicates can be expressed analogically.

In OWL, `rdfs:subPropertyOf` construct is used for expressing such semantic relationships:

Example 7 *The OWL interpretation of the relationship of the properties* `date_` *and* `date`

```
<owl:DatatypeProperty rdf:about="#date_">
    <rdfs:subPropertyOf rdf:resource="#date" />
</owl:DatatypeProperty>
```

This relationship is more vague than the relationship of equivalence. However, the relationship of “subPropertyOf” mirrors SPOT_images being conceptually more specific than VirGIS_images.

For completeness, there is an additional class in the model. `Geometry` class contains geometric elements, designed for geometry type properties description. In case that richer geometry is needed, geometry classes from existing spatial ontologies can be imported. At this time, the presented ontology is suitable for VirGIS data description. It can be enriched in case more capabilities should be needed.

5. Conclusion

Data integration is a real problem of information processing for a long time. There were already done some solving steps, whether partial solutions in particular research areas, or development towards the Semantic Web. A lot of work must be still done. The first step for in this paper proposed system was done. A new ontology describing sources and data in the VirGIS integration system was developed. Further tasks are planned: mapping expression, query rewriting, and infer mechanism and tools.

References

[1] Z. Bellahsene, "Data integration over the Web", *Data&Knowledge Engineering*, vol. 44, pp. 265–266, 2003.

[2] M.-R. Koivunen and E. Miller, "W3C Semantic Web Activity", in the proceedings of the *Semantic Web Kick/off Seminar*, Finland, 2001.

[3] B. Korte George, "The GIS (Geographic Information Systems)", *OnWord Press*, Santa Fe, 1994.

[4] O. Boucelma and F.-M. Colonna, "Mediation for Online Geoservices", in Proc. *4th International Workshop Web & Wireless Geographical Information System, W2GIS 2004*, Korea, 2004.

[5] W3C (WWW Consortium), `http://www.w3.org`.

[6] Extensible Markup Language (XML), `http://www.w3.org/XML/`.

[7] Resource Description Framework (RDF), `http://www.w3.org/RDF/`.

[8] RDF Vocabulary Description Language 1.0: RDF Schema, *W3C Recommendation*, `http://www.w3.org/TR/2004/REC-rdf-schema-20040210`, February, 2004.

[9] Web Ontology Language (OWL), `http://www.w3.org/2004/OWL`.

[10] B. DuCharme and J. Cowan, "Make Your XML RDF-Friendly", October, 2002, `http://www.xml.com/pub/a/2002/10/30/rdf-friendly.html`.

[11] SPARQL Query Language for RDF, *W3C Working Draft*, October, 2004.

[12] O. Corcho, M. Fernández-López, and A. Gómez-Pérez, "Methodologies, tools and languages for building ontologies. Where is their meeting point?", *Data & Knowledge Engineering*, vol. 46, pp. 41–64, 2003.

[13] The Protégé Ontology Editor and Knowledge Acquisition, `http://protege.stanford.edu/index.html`.

Relational Data Model with Preferences

Post-Graduate Student:
ING. RADIM NEDBAL

Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2

182 07 Prague 8

radned@seznam.cz

Supervisor:
ING. JÚLIUS ŠTULLER, CSC.

Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2

182 07 Prague 8

stuller@cs.cas.cz

Field of Study:
Mathematical Engineering
Classification: X11

Abstract

The paper proposes to extend the classical relational data model by the notion of preference realized through a partial ordering on the set of relation instances. This extension involves not only data representation but also data manipulation. As the theory of the majority of query languages for the relational model is based on the relational algebra and because of its fundamental nature, the algebra can be regarded as a basic measure of expressive power for database languages in general. To reach this expressive power in the proposed – semantically reacher – extended relational data model, the relational algebra operators need to be generalized. Simultaneously, it is desirable to preserve their usual properties. To sum up, the proposed extension of the relational model should be as minimal as possible, in the sense that the formal basis of the relational model is preserved. At the same time, the extended model should be fundamental enough to provide a sound basis for the investigation of new possible applications.

1. Introduction

Preference modelling is used in a great variety of fields. The purpose of this article is to present fundamental ideas of preference modelling in the framework of relational data model.

In this section, a brief overview of related research work and fundamentals of the proposed, extended relational model are presented. In the second section, the notion of preference and methods of its realization through ordering is introduced. In particular, order representation through $\langle P, I \rangle$ *preference structure* on attribute domains and ordering on instances of a given relation are explored. The third section discusses an effective implementation method through a generalized Hasse diagram notation. The last section summarizes the solutions pursued and the the approach potential.

1.1. Related Research Work

Recent work in AI and related fields has led to new types of preference models and new problems for applying preference structures.

Preference modelling fundamental notions as well as some recent results present Öztürk et al. [6]. The authors discuss different reasons for constructing a model of preference and number of issues that influence the construction of preference models. Information used when such models are established is analyzed, and different sources and types of uncertainty are introduced. Also, different formalisms, such as classical and nonclassical logics, and fuzzy sets, that can be used in order to establish a preference model are discussed, and different types of preference structures reflecting the behavior of a decision maker: classical, extended and valued ones, are presented. The concepts of thresholds and minimal representation are also introduced. Finally, the concept of deontic logic (logic of preference) and other formalisms associated with "compact representation of preferences", introduced for special purposes, are explored.

As ordering is inherent to the underlying data structure in database applications, Ng [5] proposes to extend the relational data model to incorporate partial orderings into data domains. Within the extended model, the partially ordered relational algebra (the PORA) is defined by allowing the ordering predicate to be used in formulae of the selection operator. The development of Ordered SQL (OSQL) as a query language for ordered databases is justified. Also, ordered functional dependencies (OFDs) on ordered databases are studied.

Nedbal [4] allows actual values of an arbitrary attribute to be partially ordered. Accordingly, relational algebra operators, aggregation functions, and arithmetic are redefined. Thus, on one side, the expressive power of the classical relational model is preserved, and, at the same time, as the new operators operate on and return ordered relations, information of preference, which is represented by a partial ordering, can be handled. Nevertheless, the redefinition of the relational operators causes loss of some of their common properties. For instance, $A = A \setminus (A \setminus B)$ does not hold. To rectify this weak point, more general concept is needed.

1.2. Extended Relational Data Model

The model proposed is a generalization of the one introduced in [4]. It extends the classical relational model both on the data representation and data manipulation levels. On the data representation level, the extension is based on incorporating an ordering into the set $\mathscr{I}(R)$ of all possible instances R^* of a relation R. Consequently, on the data manipulation level, the operators of relational algebra need to be generalized to enable handling the new information represented by the ordering. Considering the minimal set of relational algebra operators, at least, five operators: union, difference, cartesian product, selection, and projection, need to be generalized.

2. Preference on a Relation

Let us start with the following illustrative and motivating example introduced in [4]:

Example 1 (Partially ordered domain) *How could we express our intention to find employees if we prefer those who speak English to those who speak German, who are preferred to those speaking any other germanic language? At the same time, we may similarly have preference for Spanish or French speaking employees to those speaking any other romanic language. To sum up, we have the following preferences:*

A. *Germanic languages:*
   1. *English,*
   2. *German,*
   3. *other germanic languages.*

B. *Romanic languages:*
   1. *Spanish of French,*
   2. *other romanic languages.*

These preferences can be formalized by an ordering, in a general case by a partial ordering on equivalence classes. The situation is depicted in the following figure. The relation `R(`<u>`NAME`</u>`, POSITION, LANGUAGE)` *of employees is represented by a table and the above preferences by means of the standard Hasse diagram notation.*

Marie is preferred to David as she speaks English and David speaks "just" German. Analogically, Patrik is preferred to Andrea due to his knowledge of French. However, Patrik and David, for instance, are "incomparable" as we have expressed no preference order between German and French. Similarly, Roman is "incomparable" to any other employee as Russian is in the preference relation with no other language. □

Dominik	President	English
Marie	Manager	English
David	Manager	German
Petr	Manager	Swedish
Adam	Manager	German
Filip	Programmer	Dutch
Martina	Programmer	English
Patrik	Programmer	French
Rudolf	Programmer	Italian
Ronald	Programmer	Spanish
Andrea	Programmer	Portuguese
Roman	Programmer	Russian

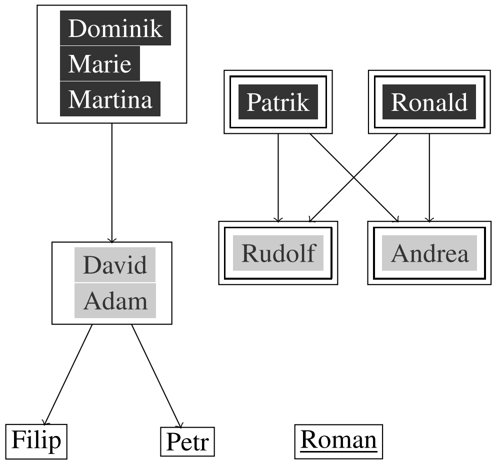


Figure 1: Partially ordered relation

Remark 1 *The ordering representing the preference from the above example can be formally described by means of $\langle P, I\rangle$ preference structure ([6]).*

Definition 1 (Preference Structure) *A preference structure is a collection of binary relations defined on the set A and such that for each couple $a, b \in A$ exactly one relation is satisfied*

Definition 2 ($\langle \mathbf{P}, \mathbf{I}\rangle$ preference structure) *A $\langle P, I\rangle$ preference structure on the set A is a pair $\langle P, I\rangle$ of relations on A such that:*

- *P is asymmetric,*
- *I is reflexive, symmetric.*

Remark 2 *Any instance, R^*, of arbitrary relation, R, with an ordering, $\preceq^{R^*}$, represented by a $\langle P, I\rangle$ preference structure, $\preceq^{R^*} = P \cup \Delta_R$, determines, through a mapping $\mathcal{P}$:*

$$\{[A; \preceq^A] | A \textit{ is arbitrary set}\} \to \mathscr{P}(\Pi(A)),$$

a set,

$$\mathcal{P}([R^*; \preceq^{R^*}]) = \{R^{F*} | t_i \preceq^{R^*} t_j \Rightarrow \mu_R(t_i^F) \leq \mu_R(t_j^F)\},$$

of fuzzy instances, R^{F*}, (of R) whose tuples,

$$t_i^F = \big(a_1, \ldots, a_n, \mu_R(t_i^F)\big),$$

have membership degrees, $\mu_R(t_i^F) \in \langle 0; 1\rangle$, consistent with the ordering $\preceq^{R^*}$, i.e.

$$t_i \preceq^{R^*} t_j \Rightarrow \mu_R(t_i^F) \leq \mu_R(t_j^F) \qquad \square$$

Consider a classical unary relational operator,

$$\mathcal{O} : \mathscr{I}(R) \to \mathscr{I}(Q),$$

operating on the set, $\mathscr{I}(R)$, of all possible instances, R^*, of a relation R. Despite each R^* being assigned a preference, $\preceq^{R^*}$, the operator, $\mathcal{O}$, returns an instance, Q^*, of a resulting relation, Q, ignoring the ordering $\preceq^{R^*}$:

$$\mathcal{O} : \{[R^*; \preceq^{R^*}] \mid R^* \in \mathscr{I}(R)\} \to \mathscr{I}(Q)$$

Therefore we would like the operator, $\mathcal{O}$, to be generalized so that its result contains ordering, based on the ordering, $\preceq^{R^*}$, of the corresponding operand R^*:

$$\mathcal{O}_G : \{[R^*; \preceq^{R^*}] \mid R^* \in \mathscr{I}(R)\} \to \{[Q_G; \preceq^{Q_G}]_1 \dots [Q_G; \preceq^{Q_G}]_n\}$$

Specifically, we would like the generalized operator, $\mathcal{O}_G$, to be consistent with respect to remark 2, i.e. to return a result, $\mathcal{O}_G([R^*; \preceq^{R^*}]) = [Q_G; \preceq^{Q_G}]$, that determines a set,

$$\mathcal{P}([Q_G; \preceq^{Q_G}]) = \{\mathcal{O}_{\mathcal{F}}(R^{F*}) \mid R^{F*} \in \mathcal{P}([R^*; \preceq^{R^*}])\},$$

containing all the fuzzy instances, $Q^{F*} = \mathcal{O}_{\mathcal{F}}(R^{F*})$, we obtain if we employ a fuzzy propositional calculi equivalent, $\mathcal{O}_{\mathcal{F}}$, of the operator $\mathcal{O}$ to fuzzy instances $R^{F*} \in \mathcal{P}([R^*; \preceq^{R^*}])$. Moreover, this consistence should be independent of a t-norm related to the fuzzy propositional calculi, $\mathcal{F}$, employed. Observe that Q_G is generally a set $Q_G \subseteq \mathscr{I}(Q)$. In brief, we are looking for such a generalized operator, O_G, that the mapping $\mathcal{P}$ is a homomorphism from algebra

$$\big(\{[R^*; \preceq^{R^*}] \mid R^* \in \mathscr{I}(R)\} \cup \{[Q_G; \preceq^{Q_G}] \mid Q_G \subseteq \mathscr{I}(Q)\}; \mathcal{O}_G\big)$$

into algebra

$$\big(\mathscr{I}_F(R) \cup \mathscr{I}_F(Q); \mathcal{O}_{\mathcal{F}}\big)$$

for any t-norm.

Intuition suggests considering tuples according to their preferences. That is to say, we always take into account the more preferred tuples ahead of those with lower preference. In addition, the other tuples that are explicitly not preferred less should also be taken into consideration. To sum up, with each tuple, t_i, we take into account a set, S_{t_i}, containing this tuple and all the tuples with higher preference:

$$S_{t_i} = \{t \mid t \in R^* \ \wedge \ t_i \preceq^{R^*} t\}$$

Then the relational operator, $\mathcal{O}$, is applied to all the elements, S_j, of $S = \{S_j = \cup_{t_i \in R^*_k} S_{t_i} | R^*_k \subseteq R^*\}$. Finally, the order, $\preceq^{Q_G}$, on the set

$$\{\mathcal{O}(S_j) | S_j \in S\} \subseteq \mathscr{I}(Q)$$

is to be determined.

Example 2 *Consider a set* $\{[\mathcal{O}(S_i); S_i] | S_i \in S\} \subseteq \mathscr{I}(Q)$ *with a relation,* $\sqsubseteq$*, implied by preference,* $\preceq^{R^*}$*, on* R^* *through the inclusion relation on* S*:*

$$[\mathcal{O}(S_i); S_i] \sqsubseteq [\mathcal{O}(S_j); S_j] \Leftrightarrow S_i \subseteq S_j$$

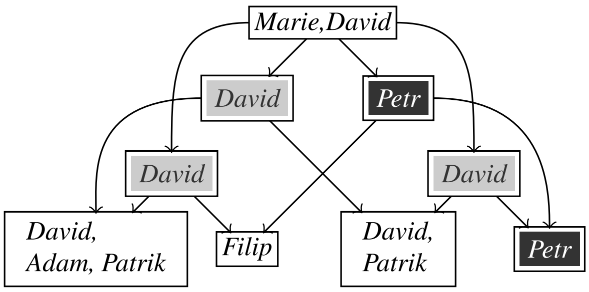


Figure 2: Projection $\{[\mathcal{O}(S_i); S_i; \sqsubseteq] | S_i \in S\}$ on $\{[\mathcal{O}(S_i); \sqsubseteq] | S_i \in S\}$

Notice that $\mathcal{O}$ generally is not an injection. In other words, $\mathcal{O}(S_i) = \mathcal{O}(S_j)$ for some $S_i \neq S_j$. In particular, $\mathcal{O}(S_i) = \mathcal{O}(S_j) =$ Petr and $\mathcal{O}(S_k) = \mathcal{O}(S_l) = \mathcal{O}(S_m) =$ David. To get an ordering, we need to resolve the duplicities:

- *Firstly, as the occurrences of "Petr" are in the relation $\sqsubseteq$, we drop the less "preferred" one.*
- *In the case of the triplet of occurrences of "David", we are unable to determine the one with the highest "preference". Nevertheless, notice that:*
  - *The set {Marie, David} is preferred to any of the occurrences of "David". In other words, whichever the most preferred occurrence of "David" is, it is less preferred then the set {Marie, David}.*
  - *There is a unique occurrence of "Filip", for which we can find an occurrence of "David" with a higher preference. In other words, whichever the most preferred occurrence of "David" is, it is preferred more then the occurrence of "Filip". The same rationale applies for the sets {David, Adam, Patrik} and {David, Patrik}.*

Thus, we get the resulting order, depicted in the following figure:

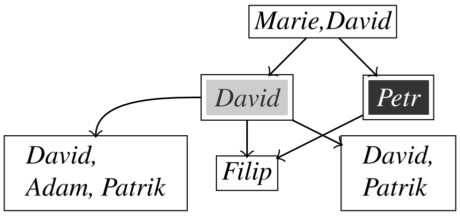


Figure 3: Ordering $\preceq^{Q_G}$ on $\{\mathcal{O}(S_i)|S_i \in S\} \subseteq \mathscr{I}(Q)$ $\square$

To sum up, the order $\preceq^{Q_G}$ on the set $\{\mathcal{O}(S_i) \mid S_i \in S\} \subseteq \mathscr{I}(Q)$ is defined as follows:

$$\mathcal{O}(S_i) \preceq^{Q_G} \mathcal{O}(S_j) \Leftrightarrow (\forall S_k \in S)\big([\mathcal{O}(S_k) = \mathcal{O}(S_i)] \Rightarrow (\exists S_l \in S)[\mathcal{O}(S_l) = \mathcal{O}(S_j) \wedge S_k \supseteq S_l]\big)$$

Approaching in this way a binary relational operator,

$$\mathcal{O} : \mathscr{I}(R) \times \mathscr{I}(R') \rightarrow \mathscr{I}(Q),$$

applied to a couple of relations, R, R', we get a set

$$\{\mathcal{O}(S_i, S'_k) \mid (S_i, S'_k) \in S \times S'\} \subseteq \mathscr{I}(Q)$$

and the order $\preceq^{Q_G}$ definition:

$$\begin{aligned}
&\mathcal{O}(S_i, S'_k) \preceq^{Q_G} \mathcal{O}(S_j, S'_l) \Leftrightarrow \\
&\qquad [\forall (S_m, S'_p) \in S \times S']\big([\mathcal{O}(S_m, S'_p) = \mathcal{O}(S_i, S'_k)] \Rightarrow \\
&\qquad\quad [\exists (S_n, S'_q) \in S \times S'][\mathcal{O}(S_n, S'_q) = \mathcal{O}(S_j, S'_l) \ \wedge S_m \supseteq S_n \ \wedge \ S'_p \supseteq S'_q]\big)
\end{aligned}$$

What are the consequences of this approach? Generally

$$\mathcal{O}(S_i) \preceq^{Q_G} \mathcal{O}(S_j) \Rightarrow \mathcal{O}(S_i) \supseteq \mathcal{O}(S_j)$$

does not hold. With respect to the relational property of *closure*, it is clear that the concept of defining preference through a $\langle P, I\rangle$ preference structure needs to be generalized.

In fact, we need to express the preference structure on powerset $\mathscr{I}(R)$ of all possible instances R^* of a relation R. This structure can be viewed through the model-theoretic approach as disjunction of conjuncts, where each conjunct, corresponding to an instance R^* of a relation R, has a given preference.

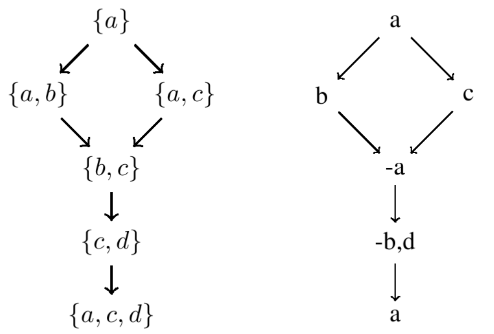
If we go further on in generalizing this structure, we get a preference structure on powerset $\mathscr{I}(DB)$ of all instances of a given database DB. It can be shown that such a structure generalizes the so-called *sure component* of M-table data structure (see Appendix) introduced by [2].

3. Sketch of Implementation

An important task to solve is the implementation of the proposed relational model. The so-called *generalized Hasse diagram* notation is suggested.

Example 3 *Consider a set $S = \{a, b, c, d\}$ and its powerset, $\mathscr{P}(S)$, with an order, $\preceq^{\mathscr{P}(S)}$, represented by means of the standard Hasse diagram notation. The order on the powerset, $\mathscr{P}(S)$, can be represented as a relation $\mathcal{R}$ on S by means of the generalized Hasse diagram notation. There is one-to-one mapping between these two representations.*

The generalization is based on the occurrences of "negative" elements, i.e. elements with a dash in front of them. Going through the diagram arrow-wise, they cancel a precedent occurrence of their "positive" equivalents (see figure 4). Moreover, all the elements depicted in the diagram are preferred to those that are absent in the diagram.



(a) Relation $\preceq^{\mathscr{P}(S)}$ on $\mathscr{P}(S)$ (b) Relation $\mathcal{R}$ on S

Figure 4: Standard and generalized Hasse diagram notation

Employing the generalized Hasse diagram notation, it is possible to develop effective algorithms for proposed, generalized relational algebra operations. Their description, however, is beyond the scope of this article. Their complexity is studied.

4. Conclusion

Methods of preference realization through $\langle P, I\rangle$ *preference structure* on attribute domains and through ordering on instances of a given relation have been discussed. Using the second approach, it has been proposed generalizing relational algebra operators in compliance with intuition. Also, a relationship of the proposed model with *M-table data structure* has been mentioned. Finally, the *generalized Hasse diagram* notation has been introduced as a means of effective implementation of the proposed model.

The proposed generalization of relational operators is necessary for a user of DBMS to be able to handle new information represented by preference. In the same way, the aggregation functions and arithmetics can be generalized.

It is possible to show that associativity and commutativity of the original union, product, restrict, and project operators are retained. Specifically for the generalized restrict operator, the following equivalences, which hold for the classical restrict operator, are retained:

$$\begin{aligned} R(\varphi_1 \vee \varphi_2) &\equiv R(\varphi_1) \cup R(\varphi_2) \\ R(\varphi_1 \wedge \varphi_2) &\equiv R(\varphi_1) \cap R(\varphi_2) \\ R(\neg\varphi) &\equiv R \setminus R(\varphi) \end{aligned}$$

Using the proposed approach, other relational operators (intersect, join, and divide), also, retain the usual properties of their classical relational counterparts:

$$\begin{aligned} R \cap S &\equiv R \setminus (R \setminus S) \\ R \div S &\equiv R[A-B] \setminus \big((R[A-B] \times S) \setminus R\big)[A-B] \\ R \bowtie S &\equiv (R \times S)(\varphi)[A] \end{aligned}$$

With respect to retaining of the above properties and equivalencies, we can conclude that the expressive power of the ordinary relational data model is maintained, and, at the same time, as the new operators operate on and return ordered relations, new information of preference represented by an ordering can be handled. This results in the ability to retrieve more accurate data.[1]

Appendix

Definition 3 (M-table) *An M-table scheme, MR, is a finite list of relation schemes $\langle R_1, \ldots, R_k\rangle, k \geq 1$, where k is referred to as the order of the M-table. An M-table over the M-table scheme, $MR = \langle R_1, \ldots, R_k\rangle$, is a pair $T = \langle T_{sure}, T_{maybe}\rangle$ where*

$$T_{sure} \subseteq \{(t_1, \ldots, t_k) \mid (\forall i)[1 \leq i \leq k \Rightarrow t_i \in \mathscr{I}(R_i)] \wedge (\exists i)[1 \leq i \leq k \wedge t_i \neq \emptyset]\}$$

$$T_{maybe} \in \{(r_1, \ldots, r_k) \mid (\forall i)[1 \leq i \leq k \Rightarrow r_i \in \mathscr{I}(R_i)]\},$$

Remark 3 *We can associate k predicate symbols, $\tilde{R}_1, \ldots, \tilde{R}_k$, with an M-table of order k. An M-table consists of two components.*

- *Sure component, which consists of mixed tuple sets, whose elements*

  $$\big\{\{t_1^1, \ldots, t_{n_1}^1\}, \ldots, \{t_1^k, \ldots, t_{n_k}^k\}\big\}$$

  represent definite and indefinite kind of information and correspond to the following logical formula

  $$[\tilde{R}_1(t_1^1) \vee \ldots \vee \tilde{R}_1(t_{n_1}^1)] \vee \ldots \vee [\tilde{R}_k(t_1^k) \vee \ldots \vee \tilde{R}_k(t_{n_k}^k)]$$

  That is, the sure component can be viewed as a conjunction of disjunctive formulas.

- *Maybe component, which consists of maybe tuples, representing uncertain information. They may or may not correspond to the truth of the real world. Some of them may have appeared in the past in mixed tuple sets and there is more reason to expect them to be the truth of the real world than others that have not been mentioned anywhere.*

[1] To my best knowledge, there is no similar study described in the literature.

References

[1] M. Ehrgott, J. Figuiera, and X. Gandibleux, editors. *State of the Art in Multiple Criteria Decision Analysis*. Springer Verlag, Berlin, 2005.

[2] K.-C. Liu and R. Sunderraman. A Generalized Relational Model for Indefinite and Maybe Information. *IEEE Transaction on Knowledge and Data Engineering*, 3(1):65–77, March 1991.

[3] R. Nedbal. Fuzzy database systems – concepts and implementation. Master's thesis, Czech Technical University, Faculty of Nuclear Sciences and Physical Engineering, Prague, June 2003.

[4] R. Nedbal. Relational databases with ordered relations. *Logic Journal of the IGPL*, 2005. Yet not published.

[5] W. Ng. An extension of the relational data model to incorporate ordered domains. *ACM Transactions on Database Systems*, 26(3):344–383, September 2001.

[6] M. Öztürk, A. Tsoukiàs, and P. Vincke. *Preference Modelling*, pages 27–72. In , [1], 2005.

List of Symbols

$\mathscr{P}(A)$	powerset of A,
$\Pi(R^*)$	$\{R_F^* \mid R_F^*$ is a fuzzy subset of $R^*\}$,
$\mathscr{I}(R)$	set of all possible instances, R^*, of a relation R,
$\mathscr{I}_F(R)$	set of all possible fuzzy instances, R^{F*}, of a relation R,
Δ_R	$\{(t,t')\mid t \in R\}$

Core problem v úlohách nejmenších čtverců

doktorand:
ING. MARTIN PLEŠINGER
Fakulta mechatroniky a mezioborových inženýrských studií
Technická universita Liberec
Hálkova 6
461 17 Liberec 1
martin.plesinger@vslib.cz

školitel:
PROF. ING. ZDENĚK STRAKOŠ, DRSC.
Ústav informatiky AV ČR
Pod Vodárenskou věží 2
182 07 Praha 8
strakos@cs.cas.cz

obor studia:
Vědecko-technické výpočty
číselné označení: M6

Tato práce byla podpořena grantem národního programu výzkumu “Informacni společnost”, č. 1ET400300415.

Abstrakt

Příspěvek stručně pojednává o core problému v úlohách lineárních nejmenších čtverců. Velmi zběžně se podíváme na klasickou úlohu nejmenších čtverců a její řešení. Dále se zaměříme na řešení úplného problému nejmenších čtverců a na komplikace, které mohou při řešení nastat.

Na úplný problém nejmenších čtverců se podíváme z poněkud odlišného úhlu, což povede k přirozené formulaci core problému. Ukážeme souvislost mezi řešením core problému a obvyklým řešením úplného problému nejmenších čtverců.

1. Úvod

V mnoha problémech matematických, fyzikálních i technických potřebujeme řešit soustavy lineárních algebraických rovnic (k popisu soustav budeme používat obvyklý maticový zápis). Je-li taková soustava čtvercová a regulární, máme k dispozici řadu metod k jejímu řešení. Často, například ve statistických aplikacích, se ale setkáváme s obecnějšími soustavami. Matice soustavy může být obecně obdélníková, hovoříme o nedourčených a přeurčených soustavách. Matice, ať už čtvercová nebo obdélníková, nemusí mít plný soupcový a/nebo řádkový rank. Vektor pravé strany obecně může ale nemusí ležet v oboru hodnot matice, může tedy ležet vně podprostoru generovaného sloupci matice.

Uvažujme tedy obecnou soustavu

$$Ax \approx b, \quad A \in \mathcal{R}^{n\times m}, \quad b \in \mathcal{R}^n, \quad \text{přičemž} \quad r \equiv \operatorname{rank}(A) \leq \min\{m,n\}, \quad m \lesseqqgtr n \tag{1}$$

a platí buď $b \notin \mathcal{R}(A)$ (*nekompatibilní systém*) nebo $b \in \mathcal{R}(A)$ (*kompatibilní systém*), obvykle předpokládáme $b \neq \emptyset$, v opačném případě je řešení triviální.

2. Klasické řešení, úplný problém nejmenších čtverců

Za klasická bychom např. mohli označit řešení problému (1) pomocí *soustavy normálních rovnic* a pomocí *rozšířené soustavy rovnic*

$$A^T A x = A^T b, \quad \text{respektive} \quad \begin{bmatrix} I & A \\ A^T & \emptyset \end{bmatrix} \begin{bmatrix} -g \\ x \end{bmatrix} = \begin{bmatrix} b \\ \emptyset \end{bmatrix},$$

kde $g = Ax - b$ je reziduum. Obecnou soustavu (1) jsme převedli na soustavu se čtvercovou, za jistých předpokladů regulární maticí, navíc s vektorem pravé strany ležícím v oboru hodnot matice soustavy. Předpokládáme, že takovou soustavu umíme řešit, viz například [1]. Řešení takto získané je řešením minimalizační úlohy

$$Ax = b + g, \quad \min_{g,x} \|g\|. \tag{2}$$

Minimalizujeme residuum g, vektor $x \perp \mathcal{N}(A)$, který nazýváme *řešení ve smyslu nejmenších čtverců minimální v normě*, je jednoznačně určen. Minimalizační úloha (2) se nazývá *problém nejmenších čtverců – LS*.

Při hledání řešení (2) v podstatě hledáme nejmenší možnou modifikaci vektoru b (pravé strany, vektoru pozorování, odezvy systému, ...). Připouštíme tak, že vektor b obsahuje chyby, ale zcela ignorujeme možnost, že chyby obsahuje i matice A (model). To je jistý nedostatek řešení (1) pomocí LS.

Chyby může obsahovat vektor b a/nebo matice A, popřípadě mohou být chyby obsažené v b a v A v nějakém vztahu. Podrobnější analýza, viz [5], ukazuje, že významná úloha, jejímuž řešení se pokusíme porozumět, je *úplný problém nejmenších čtverců – TLS*,

$$(A + E)x = b + g, \quad \min_{g,E,x} \|[g|E]\|_F. \tag{3}$$

V dalším textu budeme tedy předpokládat, že chyby obsahuje celá soustava, celá matice $[b|A]$.

3. Analýza Goluba a van Loana

Pro jednoduchost budeme v této sekci uvažovat (1) s maticí A, která má plný sloupcový rank, navíc budeme předpokládat $b \notin \mathcal{R}(A)$.

Z analýzy Goluba a van Loana [2] vyplývá, že problém (3) lze za výše uvedených předpokladů a za jisté níže uvedené podmínky (6) přeformulovat a maticově zapsat

$$\left[\; b + g | \; A + E \; \right] \begin{bmatrix} -1 \\ x \end{bmatrix} = \emptyset. \tag{4}$$

Vidíme, že matice $[b + g|A + E]$ má netriviální nulový prostor (jádro), ve kterém leží vektor s nenulovou první komponentou.

Řešme tedy úlohu (3). Nechť ekonomický singulární rozklad matice $[b|A]$ je

$$[b|A] = U_q \Sigma_q V_q^T = \sum_{i=1}^{q} u_i \sigma_i v_i^T, \quad \text{přičemž} \quad q \equiv \operatorname{rank}([b|A]) \leq \min\{m + 1, n\}.$$

Obecně platí $r \leq q \leq r + 1$, kde r je hodnost matice A, z předpokladů na začátku sekce ovšem vyplývá $r = m < n$, $q = m + 1 \leq n$, tedy $r < q = r + 1$. Minimální perturbace $[g|E]$ matice $[b|A]$ taková, abychom dostali matici s netriviálním jádrem, je

$$[g|E] \equiv -u_q \sigma_q v_q^T, \tag{5}$$

vektor ležící v jádru matice je $v_q = (\nu_1, \ldots, \nu_{m+1})^T = (\nu_1, w^T)^T$. Za předpokladu (6) je $\nu_1 \neq 0$, viz [2], zřejmě platí

$$\begin{bmatrix} -1 \\ x \end{bmatrix} \equiv -\frac{1}{\nu_1} v_q, \quad \text{vektor} \quad x \equiv -\frac{1}{\nu_1} w$$

je řešením minimalizační úlohy (3), nazýváme ho *řešení ve smyslu TLS*, výše popsaný postup je *algoritmem Goluba a van Loana – AGVL*. Toto řešení můžeme také nazývat, zejména v kontextu další sekce, *generické řešení*.

Obtíže nastanou pokud $\nu_1 = 0$, vektorově $e_1^T v_q = 0$. Minimální petrurbace (5) snižující hodnost matice $[b|A]$ sice lze sestrojit, ale řešení úlohy (3) neexistuje, místo minima existuje pouze infimum. Lze ukázat, že pro některá A, b ztrácí formulace (3) problému TLS zcela smysl, viz příklad v [2, str. 884].

Ona "jistá podmínka", za které lze problém (3) přeformulovat v (4), která zajistí $\nu_1 \neq 0$, je

$$\sigma_r \equiv \sigma_{\min}(A) > \sigma_{\min}([b|A]) \equiv \sigma_q. \tag{6}$$

Je-li (6) splněna, formulace úlohy (3) má vždy smysl a problém lze řešit pomocí AGVL. Bohužel tato podmínka je pouze podmínkou postačující, nikoliv podmínkou nutnou!

4. Negenerické řešení

Sabine Van Huffel a Joos Vandewalle [3] navazují na předchozí analýzu, ale zavádějí kvalitativně odlišné řešení aproximačního problému (1). Toto řešení je v jistém smyslu zobecněním řešení TLS problému (3).

Tentokrát ovšem budeme vycházet ze zápisu (4) (uvažujeme zcela obecný problém (1)). Chceme získat nejmenší petrubaci $[g|E]$ takovou, aby byla splněna rovnost (4), hledáme matici, v jejímž jádru leží vektor s nenulovou první komponentou. Nejmenší taková petrubace $[g|E]$ je zřejmě

$$[g|E] \equiv -u_s \sigma_s v_s^T, \quad v_s = \begin{bmatrix} \mu_1 \\ z \end{bmatrix}, \tag{7}$$

kde σ_s je nejmenší singulární číslo takové, že $\mu_1 \neq 0$. Řešení

$$x \equiv -\frac{1}{\mu_1} z$$

nazýváme *negenerické řešení TLS*, postup nazýváme *rozšířením Van Huffel a Vandewalle – EVHV*.

Negenerické řešní TLS odpovídá řešení původní minimalizační úlohy (3) s rozšiřující podmínkou

$$(A+E)x = b + g, \quad \min_{g,E,x} \|[g|E]\|_F \quad \wedge \quad [g|E][v_{s+1}, \ldots, v_q] = \emptyset. \tag{8}$$

Negenerické řešení vždy existuje (pro libovolná A, b z (1)) a v případě, že je splněna podmínka (6), je identické s řešením nalezeným pomocí AGVL, podrobněji viz [3]. Toto řešení je tedy zobecněním (rozšířením) generického řešení na data nesplňující podmínku (6). Ve skutečnosti však hledáme negenerické řešení minimální v normě, viz [3], zde to pro jednoduchost vynecháváme.

5. Odlišný úhel pohledu

Uvažujme aproximační problém (1) a ortogonální matice P, Q odpovídajících rozměrů. Transformovaný problém

$$\tilde{A}\tilde{x} \equiv \left[P^T A Q\right] \left[Q^T x\right] \approx \left[P^T b\right] \equiv \tilde{b}, \qquad P^{-1} = P^T, \;\; Q^{-1} = Q^T \tag{9}$$

má, až na transformaci $x = Q\tilde{x}$, stejné generické, resp. negenerické řešení jako problém původní (neboť Frobeniova norma i singulární rozklad jsou ortogonálně invariantní). Předpokládejme, že transformovaný problém má následující strukturu

$$P^T[b|AQ] = [\tilde{b}|\tilde{A}] = \left[\begin{array}{c||c|c} b_1 & A_{11} & \emptyset \\ \hline \emptyset & \emptyset & A_{22} \end{array}\right]. \tag{10}$$

Původní problém $[b|A]$ se rozpadl na dva podproblémy $[b_1|A_{11}]$ a $[\emptyset|A_{22}]$, z nichž druhý má zřejmě jediné smysluplné řešení $x_2 = \emptyset$. Intuice napoví, že řešení původního problému je bezpodmínečně

$$x = Q \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = Q \begin{bmatrix} x_1 \\ \emptyset \end{bmatrix}, \tag{11}$$

kde x_1 je řešení prvního podproblému. Bez újmy na obecnosti, jak uvidíme později, můžeme předpokládat, že první podproblém $A_{11}x_1 \approx b_1$ vyhovuje (6) (je řešitelný pomocí AGVL a x_1 je jeho generickým řešením).

Abychom nahlédli, kde se skrývá obtíž při řešení problému pomocí AGVL, budeme předpokládat

$$\sigma_q = \sigma_{\min}(A_{22}) < \sigma_{\min}([b_1|A_{11}]),$$

intuitivní řešení (11) se tím nijak nezmění. Pokusme se celý problém vyřešit pomocí AGVL (hned si všimněme, že není splněna (ovšem postačující, ne nutná) podmínka (6)). Platí následující rovnost

$$\left[\begin{array}{c||c|c} b_1 & A_{11} & -r_1\theta^{-1}v_q^T \\ \hline \emptyset & \emptyset & A_{22} - u_q\sigma_q v_q^T \end{array}\right] \begin{bmatrix} -1 \\ \tilde{x} \end{bmatrix} = \emptyset, \quad \text{kde} \quad \tilde{x} = \begin{bmatrix} z \\ v_q\theta \end{bmatrix} \quad \text{a} \quad x = Q \begin{bmatrix} z \\ v_q\theta \end{bmatrix}, \tag{12}$$

přičemž u_q, v_q jsou levý, resp. pravý singulární vektor odpovídající singulárnímu číslu σ_q ze singulárního rozkladu bloku A_{22}, z je libovolně zvolený vektor a $r_1 \equiv A_{11}z - b_1$, $\theta > 0$ je kladné číslo. Perturbace $[g|E]$ původního systému $[b|A]$ tak je

$$[g|E] = P \left[\begin{array}{c||c|c} \emptyset & \emptyset & -r_1\theta^{-1}v_q^T \\ \hline \emptyset & \emptyset & -u_q\sigma_q v_q^T \end{array}\right] \left[\begin{array}{c|c} 1 & \emptyset \\ \hline \emptyset & Q^T \end{array}\right].$$

Pokud se pokusíme nalézt minimální perturbaci, narazíme na problém. Zřejmě pro $\theta \to +\infty$

$$\|[g|E]\|_F = \sqrt{\|r_1\|^2\theta^{-2} + \sigma_q^T} \quad \longrightarrow \quad \sigma_q = \sigma_{\min}(A_{22})$$

a zároveň

$$\|x\| = \|\tilde{x}\| = \left\| \begin{bmatrix} z \\ v_q\theta \end{bmatrix} \right\| \quad \longrightarrow \quad +\infty.$$

Minimum tedy neexistuje, existje pouze infimum. Narazili jsme právě na případ, kdy formulace (3) TLS problému zcela postrádá smyslu (ostatně snadno nahlédneme, že pravý singulární vektor odpovídající σ_q v rozkladu matice $[\tilde{b}|\tilde{A}]$ musí mít první složku nulovou). Řešení (3) v tomto případě neexistuje, a tak ani x řešící (12) nemůže být ani vzdálenou aproximací řešení (3). Nedosti na tom, x řešící (12) je, krom libovolně zvolených komponent z a θ, tvořeno pravým singulárním vektorem bloku A_{22}, který jsme v intuitivním přístupu celý zanedbali.

Důležité je následující pozorování, jež nám bude motivací při formulování core problému:

- Pokud se nám původní problém podaří transformovat na (10) a řešení budeme hledat ve tvaru (11), zcela potlačíme vliv dat obsažených v bloku A_{22}. Jinými slovy: z problému odfiltrujeme komponenty související se singulárními čísly $\sigma_i(A_{22})$ bloku A_{22}.

  Tento blok není při intuitivním přístupu vůbec nutný k řešení problému, informace v něm obsažené jsou zcela irelevantní a s řešením problému, tedy i s problémem samotným, vůbec nesouvisí.

- Řešíme-li původní problém pomocí EVHV dospějeme k řešení shodnému s intuitivním řešením (11) (všechny pravé singulární vektory z rozkladu matice $[\tilde{b}|\tilde{A}]$ odpovídající singulárním číslům bloku A_{22} mají zřejmě nulovou první složku). Pomocí EVHV odfiltrujeme část informace obsažené v bloku A_{22}, ovšem odfiltrujeme pouze ta data, která souvisejí se singulárnými čísly $\sigma_i(A_{22}) < \sigma_s$ (tedy pro $i > s$), viz (8).

6. Core problem

Vhodné by bylo pro řešení (1) využít pouze informace nutné a postačující k řešení. Naší snahou bude odfiltrovat veškerou informaci, která s problémem nesouvisí a na jeho řešení nemá vliv. V dalším textu budeme předpokládat $A^T b \neq 0$, tj. vektor pravé strany není ortogonální na podprostor generovaný sloupci matice A (v opačném případě existuje pouze negenerické řešení a je triviální $x = \emptyset$, přičemž $[g|E] = [-b|\emptyset]$; předpoklad přirozeně zahrnuje i případ, kdy b je identicky nulový vektor).

Budeme hledat ortogonální matice P, $P^{-1} = P^T$, Q, $Q^{-1} = Q^T$ transformující původní problém na problém se strukturou (10), navíc budeme požadovat aby blok A_{22} měl maximální možnou dimenzi (a blok $[b_1|A_{11}]$ minimální).

Definice 1 *Podproblém $A_{11}x_1 \approx b_1$ z rozkladu (10) nazveme* **core problem** *v aproximačním problému $[b|A]$ v případě, že $[b_1|A_{11}]$ má minimální dimenzi (a A_{22} maximální dimenzi).*

Pokusíme se core problém nalézt. Nechť

$$A = U\Sigma V^T = U \left[\begin{array}{c|c} \Sigma_r & \emptyset \\ \hline \emptyset & \emptyset \end{array} \right] V^T = \sum_{i=1}^{r} u_i \sigma_i v_i^T, \quad \Sigma_r = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$$

je singulární rozklad matice A. Zaveďme pomocné značení $U = [U_1|U_2]$, kde $U_1 = (u_1, \ldots, u_r)$, $U_2 = (u_{r+1}, \ldots, u_n)$, a analogicky $V = [V_1|V_2]$, kde $V_1 = (v_1, \ldots, v_r)$, $V_2 = (v_{r+1}, \ldots, v_m)$. Platí

$$U^T[b|AV] = \left[\begin{array}{c||c|c} c & \Sigma_r & \emptyset \\ \hline d & \emptyset & \emptyset \end{array} \right]. \tag{13}$$

Vektory c, d mohou obsahovat nulové prvky, pomocí ortogonálních transformací se budeme snažit problém upravit tak, aby tyto vektory obsahovaly maximum nulových prvků (tím nalezneme core problém). Nejprve ortogonální maticí H_{22} (Householderova reflexe) modifikujeme vektor d tak, že $H_{22}^T d = e_1\delta$, $\delta = \|d\|$. Podmatici U_2 v rozkladu (13) nahradíme maticí U_2H_{22}.

Nyní budeme upravovat vektor $c = (\gamma_1, \ldots, \gamma_r)$. Uvažujme $\sigma_i = \ldots = \sigma_j$ singulární čísla matice A, jinak řečeno, σ_i je singulární číslo s násobností $j - i + 1$, přičemž $j \geq i$. Pomocí ortogonální matice H_{ij} (Householderova reflexe) transformujeme jim odpovídající podvektor $c_{ij} = (\gamma_i, \ldots, \gamma_j)$ tak, že $H_{ij}^T c_{ij} = e_1\gamma_{ij}$, $\gamma_{ij} = \|c_{ij}\|$. Označme H_{11} ortogonální, blokově diagonální matici, která má na diagonále matice H_{ij} (signulárním číslům s násobností jedna tedy odpovídají jednotky na diagonále). Touto transformací získáme na místě vektoru c vektor s maximálním možným počtem nulových prvků. Nalezneme permutaci P_{11}^T řádků matice $H_{11}^T[c|\Sigma_r H_{11}]$ tak, že (lidově řečeno) řádky s nenulovou první složkou seřadíme pod sebe a všechny řádky začínající nulou odsuneme dolů. V rozkladu (13) nahradíme podmatici U_1 maticí $U_1H_{11}P_{11}$ a podmatici V_1 maticí $V_1H_{11}P_{11}$.

V případě, že $\delta \neq 0$, provedeme ještě permutaci řádků P_0^T tak, že tento řádek zařadíme přímo pod ostatními řádky s nenulovou první složkou, zřejmě $\delta \neq 0$ tehdy a jen tehdy, je-li původní problém nekompatibilní, tedy když $b \notin \mathcal{R}(A)$. Získáme tak rozklad

$$P^T[b|AQ] \equiv \left(P_0^T[U_1H_{11}P_{11}|U_2H_{22}]^T \right) \left[b|A \left([V_1H_{11}P_{11}|V_2] \right) \right] = \left[\begin{array}{c||c|c} \tilde{c} & \Sigma_1 & \emptyset \\ \delta & \emptyset & \emptyset \\ \hline \emptyset & \emptyset & \Sigma_2 \end{array} \right], \tag{14}$$

kde Σ_1 obsahuje pouze vzájemně různá signulární čísla matice A (ovšem obecně ne všechna), Σ_2 obsahuje všechna ostatní singulární čísla matice A, všechna opakování singulárních čísel a navíc (pro přehlednost zápisu) obsahuje nulová singulární čísla. Tím jsme problém transformovali na blokovou strukturu (10).

Věta 1 *Existuje taková ortogonální transformace (9), tedy takové ortogonální matice P, Q, že blok A_{11}, resp. A_{22} z rozkladu (10) má minimální, resp. maximální možnou dimenzi přes všechny ortogonální transformace vedoucí na danou strukturu. Blok A_{11} nemá žádné opakující se a ani nulové singulární číslo a blok A_{22} obsahuje všechna opakování singulárních čísel (redundance), všechna nerelevantní data (singulární čísla jímž odpovídající levé singulární podprostory jsou ortogonální na vektor b) a všechna nulová singulární čísla.*

Podproblém $A_{11}x_1 \approx b_1$ je core problém a je vždy řešitelný pomocí AGVL a podproblém $A_{22}x_2 \approx \emptyset$ má jediné smysluplné řešení $x_2 = \emptyset$. Řešení původního problému je

$$x = Q \begin{bmatrix} x_1 \\ \emptyset \end{bmatrix}. \tag{15}$$

Důkaz věty 1 byl z části proveden v přechozím textu, podrobněji viz [4].

Věta 2 *Matice A_{11} z věty 1 je čtvercová matice $\mathcal{R}^{p \times p}$, resp. obdélníková matice $\mathcal{R}^{p+1 \times p}$ tehdy a jen tehdy, když vektor b má právě p nenulových projekcí do levých singulárních podprostorů odpovídajících různým singulárním číslům matice A a zároveň $b \in \mathcal{R}(A)$, resp. $b \notin \mathcal{R}(A)$.*

Věta 3 *Podproblém $A_{11}x_1 \approx b_1$ z rozkladu (10) je kompatibilní tehdy a jen tehdy, když je kompatibilní celý problém $Ax \approx b$, tedy*

$$b \in \mathcal{R}(A) \quad \Longleftrightarrow \quad b_1 \in \mathcal{R}(A_{11}).$$

Jinými slovy: v kompatibilním případě je matice $A_{11} \in \mathcal{R}^{p \times p}$ z věty 1 čtvercová a regulární.

Důkazy obou vět 2, 3 vyplývají z textu, z konstrukce rozkladu (14), podrobněji opět viz [4].

7. Nalezení core problému

Ortogonální matice P, Q z věty 1 dokážeme nalézt přímo, transformací matice $[b|A]$ na horní bidiagonální tvar (Householderovy reflexe, nebo Lanczos-Golub-Kahanova bidiagonalizace). Důkaz využívající singulární rozklady jednotlivých bloků rozkladu (10) nalezneme v článku [4].

S výhodou využijeme faktu, že bidiagonalizaci provádíme postupně. Díky tomu můžeme proces zastavit v momentě, když dojde k oddělení obou podproblémů, jinak řečeno: blok A_{22} nemusí být bidagonalizován. K oddělení obou podproblémů dojde:

- V kompatibilním případě $b \in \mathcal{R}(A)$

$$[b_1|A_{11}] = \left[\begin{array}{c|cccc} \beta_1 & \alpha_1 & & & \\ & \beta_2 & \alpha_2 & & \\ & & \ddots & \ddots & \\ & & & \beta_p & \alpha_p \end{array}\right] \in \mathcal{R}^{p \times p+1}, \quad \alpha_i\beta_i \neq 0, \quad i = 1 \dots p.$$

  pokud $\beta_{p+1} = 0$ nebo $p = n$. Matice A_{11} je čtvercová dimenze p, nesingulární. Core problém $A_{11}x_1 = b_1$ je kompatibilní.

- V nekompatibilním případě $b \notin \mathcal{R}(A)$

$$[b_1|A_{11}] = \left[\begin{array}{c|cccc} \beta_1 & \alpha_1 & & & \\ & \beta_2 & \alpha_2 & & \\ & & \ddots & \ddots & \\ & & & \beta_p & \alpha_p \\ & & & & \beta_{p+1} \end{array}\right] \in \mathcal{R}^{p+1 \times p+1}, \quad \begin{array}{l} \alpha_i\beta_i \neq 0, \quad i = 1 \dots p, \\ \beta_{p+1} \neq 0. \end{array}$$

  pokud $\alpha_{p+1} = 0$ nebo $p = m$. Matice $[b_1|A_{11}]$ je čtvercová dimenze $p + 1$, nesingulární. Core problém $A_{11}x_1 \approx b_1$ je nekompatibilní.

V obou případech má matice A_{11} plný sloupcový rank a matice $[b_1|A_{11}]$ má plný řádkový rank.

8. Shrnutí a závěr

Řešení původního problému získané technikou core problému je obecně shodné s negenerickým řešením (s řešením pomocí EVHV), za předpokladu (6) je shodné s řešením generickým (pomocí AGVL). Výhodou přístupu pomocí core problému je využití pouze nutných a postačujících informací k řešení. Oprávněnost řešení (15) core problému (10), narozdíl od Golubova, van Loanova řešení podmíněného (6) a rozšířeného na negenerické řešení (8), je snadno nahlédnutelná a význam řešení (15) zcela odpovídá naší intuici. Celkový vhled do problematiky prostřednictvím core problému je jasný a zřejmý, čehož není možné dosháhnout prostřednictvím AGVL a EVHV.

Celý postup přitom není složitější než přístup pomocí AGVL a EVHV, ba právě naopak. Abychom zjistili, zda je splněna podmínka (6), potřebujeme spočítat obě singulární čísla $\sigma_{\min}(A)$ a $\sigma_{\min}([b|A])$. Teprve potom můžeme rozhodnout zda hledat generické nebo negenerické řešení. K nalezení core problému potřebujeme provést bidiagonalizaci matice $[b|A]$, ovšem neúplnou, neboť blok A_{22} nemusí být bidiagonalizován. Core problém pak lze bezpodmínečně řešit pomocí AGVL. K porovnání složitostí obou přístupů si stačí uvědomit, že bidiagonalizace je první (finitní) částí algoritmu pro výpočet singulárního rozkladu.

Kolem problematiky lineárních nejmenších čtverců řešených prostřednictvím core problému je řada důležitých, otevřených otázek:

- Jak vytvořit analogickou teorii pro úlohy s vícenásobnou pravou stranou?
- Při řešení ill-posed problémů provádíme regularizaci, hledáme minimum

$$\min\left(\|Ax-b\|+\|Lx\|\right),$$

  kde L je matice určená pro daný problém. V případě, že matice L kombinuje řešení obou podproblémů $A_{11}x_1 \approx b_1$ a $A_{22}x_2 \approx \emptyset$, nemůžeme položit $x_2 = \emptyset$.
- Celá teorie byla odvozena v přesné aritmetice. Jak se ovšem bude core problém chovat v aritmetice s konečnou přesností? Jaká je citlivost a jaké je numerické chování core problému?
- Neposledním důležitým úkolem je numericky stabilní implementace software pro řešení úloh (1) pomocí core problému.

Literatura

[1] G. Dahlquist, Å. Björck, “Numerical Mathematics and Scientific Computation”, Vol. 2, Chap. 8, Linear Least Squares Problem, Web Draft, 2005.

[2] G. H. Golub, C. F. van Loan, “An Analysis of The Total Least Squares Problem”, *Numer. Anal.*, vol 17, pp. 883–893, 1980.

[3] S. Van Huffel, J. Vandewalle, “The Total Least Squares Problem: Computational Aspects and Analysis”, SIAM Publications, Philadelphia PA, 1991.

[4] C. C. Paige, Z. Strakoš, “Core Problems in Linear Algebraic Systems”, Accepted for publications in *Matrix Anal.*, 2005.

[5] C. C. Paige, Z. Strakoš, “Scaled Total Least Squares Fundamentals”, *Numerische Mathematik*, vol. 91, pp. 117–146, 2002.

Mezinárodní nomenklatury a metatezaury ve zdravotnictví

doktorand:
MGR. PETRA PŘEČKOVÁ

EuroMISE Centrum – Kardio
Ústav informatiky AV ČR
Pod Vodárenskou věží 2

182 07 Praha 8

preckova@euromise.cz

školitel:
PROF. RNDR. JANA ZVÁROVÁ, DRSC.

EuroMISE Centrum – Kardio
Ústav informatiky AV ČR
Pod Vodárenskou věží 2

182 07 Praha 8

zvarova@euromise.cz

obor studia:
Biomedicínská informatika
číselné označení: 3918V

Článek vznikl s podporou grantu 1ET200300413 Akademie věd České republiky.

Abstrakt

Využití mezinárodních nomenklatur a metatezaurů pro kódování terminologie ve zdravotnictví je prvním nezbytným krokem interoperability heterogenních systémů zdravotních záznamů, která je základem pro sdílenou zdravotní péči vedoucí k efektivitě ve zdravotnictví, finančním úsporám i snížení zátěže pacientů. V tomto článku popisuji různé mezinárodní nomenklatury a metatezaury používané ve zdravotnictví. Hlavní důraz kladu na *Unified Medical Language System* a zejména na *UMLS Metatezaurus*, který mi nejvíce napomáhá při mapování odborné zdravotnické terminologie. Svou prací se snažím o ověření praktické použitelnosti mezinárodně používaných terminologických slovníků, tezaurů, ontologií a klasifikací a to na atributech *Minimálního datového modelu kardiologického pacienta*, *Datového standardu Ministerstva zdravotnictví České republiky* a některých vybraných modulů komerčních nemocničních informačních systémů. Článek popisuje, jakým problémům při mapování čelím a nastiňuje jejich řešení.

1. Úvod

Vymezení, pojmenování a třídění lékařských pojmů ve srovnání s ostatními přírodními vědami dosud není optimální. Dokladem je skutečnost, že pro jeden pojem existuje často více než deset synonym, chápání přesnějšího vymezení klinické jednotky (příznak, diagnóza) je v řadě oborů u jednotlivých lékařských škol rozdílné i v národním měřítku a mezinárodně uznávané konvence dosud nejsou příliš časté. Větší řád je např. v rámci botaniky a zoologie. V těchto oborech je zákonem autorská priorita, tzn. že pojmenování je platné jen podle autora, jenž popsal druh jako první. To vede k zamezení opakovaného popisu téhož druhu s různým názvem a tím i synonym.

Praktickým negativním důsledkem v lékařství je situace, kdy je například efekt nového léku nebo hodnot nové vyšetřovací metody u dané diagnózy popisován ve dvou publikacích. Pokud je chápání této diagnózy v každé z uvedených publikací poněkud posunuto a jedná se tedy o rozdílné množiny pacientů, můžeme se často setkat i s kontroverzními výsledky, což hodnotu výsledné informace samozřejmě snižuje.

Se zaváděním výpočetní techniky v lékařství se tento problém pouze prohloubil, neboť její využívání předpokládá větší jednoznačnost zadávání dat, vymezení pojmů, jejich přesné pojmenování atd., čímž se značné nedostatky projevují ještě výrazněji.

Obecně je velmi výhodné využívat v odborné terminologii pro jeden pojem vždy pouze jediný výraz. Synonyma lze sice počítač naučit, zvětšují však rozsah slovníku databáze i počet nezbytných operací, což prodlužuje komunikaci. Synonymie v odborné terminologii vede při sdělování informací navíc k nepřesnostem a nedorozumění. V současné lékařské terminologii se lze nyní setkat s řadou synonym pro jediné onemocnění.

2. Klasifikační systémy

Klasifikační systémy (klasifikace) jsou takové kódovací systémy, které jsou založeny na principu vytváření tříd. Třídy tvoří agregované pojmy, které se shodují v alespoň jednom klasifikačním atributu. Třídy klasifikace musí pokrývat úplně vymezenou oblast a nesmí se překrývat. Tvorba klasifikačních systémů a *nomenklatur* byla motivována především jejich praktickým využitím v evidenci, třídění a statistickém zpracování lékařské informace. Prvotním zájmem bylo evidovat výskyt nemocí a příčiny smrti.

2.1. ICD - International Classification of Diseases

Základy *Mezinárodní klasifikace nemocí* [1] (ICD) položil William Farr v roce 1855. V roce 1948 ji přejala *Světová zdravotnická organizace* WHO. V této době šlo již o 6. verzi. Základním nedostatkem ICD je její nižší stupeň hierarchie. ICD vyhovuje pro účely statistiky diagnóz, nikoli však pro další kódování komplexní lékařské informace, jelikož zde chybí například pojmy pro příznaky a terapii. Poslední revize se ale snaží o co nejpodrobnější klasifikace (místo první číslice je písmeno latinské abecedy, další místa jsou číslice).

V současné době se používá 10. revize ICD, která platí od roku 1994 a obsahuje 22 kapitol.

2.2. SNOMED

Akronym *SNOMED* [2] vznikl ze spojení ***S**ystematized **NO**menclature of **MED**icine*. SNOMED byl prvně publikován v roce 1965. Jedná se o detailní klinickou referenční terminologii založenou na kódování. Skládá se z 344 549 pojmů vztahujících se ke zdravotnictví a umožňuje využívat zdravotnické informace kdykoli a kdekoli je to potřeba. SNOMED poskytuje "společný jazyk", který umožňuje konzistentní způsob získávání, sdílení a shromažďování zdravotnických dat od různých klinických skupin, mezi které patří ošetřovatelství, medicína, laboratoře, lékárny i veterinární medicína. Tento klasifikační systém je používán ve více než 40 státech. SNOMED umožňuje popis jakékoli situace v medicíně pomocí 11 úrovní - dimenzí: *topografie*; *morfologie*; *funkce*; *živé organismy*; *fyzičtí činitelé*, *aktivity a síly*; *chemikálie*, *léky a biologické produkty*; *procedury*; *zaměstnání*; *sociální kontext*; *nemoci/diagnózy*; *modifikátory*. Jednotlivé pojmy jsou označovány zkratkou dimenze a 5-místným zvláštním kódem, kde je využíváno čísel 0-9 a navíc písmen A-F. Jednotlivá místa kódu směrem doprava stále zpřesňují obsah popisovaného pojmu.

2.3. MeSH

***Me**dical **S**ubject **H**eadings* (MeSH) [3] je slovník kontrolovaný *Národní lékařskou knihovnou* (NLM) v USA. Tvoří ho skupina pojmů, které hierarchicky pojmenovávají klíčová slova a tato hierarchie napomáhá při vyhledávání na různých úrovních specifičnosti. Klíčová slova jsou uspořádána jak abecedně tak hierarchicky. Na nejobecnější úrovni hierarchické struktury jsou široké pojmy jako např. "anatomie" nebo "mentální onemocnění". NLM využívá MeSH k indexování článků ze 4600 světových předních biomedicínských časopisů pro databázi *MEDLINE/PubMED*®. MeSH se využívá také pro databázi katalogizující knihy, dokumenty a audiovizuální materiály. Každý bibliografický odkaz je spojován se skupinou termínů v klasifikačním systému MeSH. Vyhledávací dotazy používají také slovní zásobu z MeSH, aby našly články na požadované téma. Specialisté, kteří MeSH slovník vytvářejí, ho průběžně aktualizují a kontrolují. Sbírají nové pojmy, které se začínají objevovat ve vědecké literatuře nebo ve vznikajících oblastech výzkumu, definují tyto pojmy v rámci obsahu existujícího slovníku a doporučují jejich přidání do slovníku MeSH. Existuje i česká verze MeSH, jejíž překlad je ale, bohužel, na poměrně nízké a těžko prakticky použitelné úrovni.

2.4. LOINC®

Klasifikační systém ***L**ogical **O**bservations **I**dentifiers, **N**ames, **C**odes* – LOINC® [4] je klinickou terminologií důležitou pro laboratorní testy a laboratorní výsledky. V roce 1999 byl LOINC® přijat organizací HL7 jako preferované kódování pro názvy laboratorních testů a klinických pozorování. Tento klasifikační systém obsahuje více než 30 000 různých termínů. Při mapování lokálních kódů různých testů na kódy LOINC napomáhá mapovací program Regenstrief LOINC Mapping Assistant (RELMA^{TM}).

2.5. ICD-O

Klasifikační systém *ICD-O* [5] je rozšířením Mezinárodní klasifikace nemocí ICD pro kódování onkologie, která byla prvně publikována WHO v roce 1976. Jedná se o čtyřdimenzionální systém, mezi jeho dimenze patří topografie, morfologie, průběh a diferenciace. Dimenze jsou určeny pro třídění morfologických typů nádorů. V současné době existuje její třetí verze.

2.6. TNM-klasifikace

TNM klasifikace [6] je klinická klasifikace maligních nádorů, která se využívá pro účely srovnávání terapeutických studií. Vychází z poznatku, že pro prognózu onemocnění je zvláště důležitá lokalizace a šíření tumoru.

2.7. DSM III.

Mezi psychiatrické nomenklatury můžeme zařadit např. *DSM III.*, která obsahuje i definice jednotlivých pojmů. Jedná se o velice propracovanou nomenklaturu. Bohužel jde o uzavřený systém bez návaznosti na další obory lékařství.

2.8. Další klasifikační systémy

V současné době existuje v medicíně více než 100 různých klasifikačních systémů. Mezi ně patří i *AI/RHEUM*; *Alternative Billing Concepts*; *Alcohol and Other Drug Thesaurus*; *Beth Israel Vocabulary*; *Canonical Clinical Problem Statement System*; *Clinical Classifications Software*; *Current Dental Terminology 2005* (CDT-5); *COSTAR*; *Medical Entities Dictionary*; *Physicians' Current Procedural Terminology*; *International Classification of Primary Care*; *McMaster University Epidemiology Terms*; *Physicians' Current Procedural Terminology*; *CRISP Thesaurus*; *COSTART*; *DSM-III-R*; *DSM-IV*; *DXplain*; *Gene Ontology*; *HCPCS Version of Current Dental Terminology 2005*; *Healthcare Common Procedure Coding System*; *Home Health Care Classification*; *Health Level Seven Vocabulary*; *Master Drug Data Base*; *Medical Dictionary for Regulatory Activities Terminology* (MedDRA); *MEDLINE*; *Multum MediSource Lexicon*; *NANDA nursing diagnoses: definitions & classification*; *NCBI Taxonomy* a mnoho dalších.

3. Prostředky pro sdílení informací z více zdrojů

Rostoucí počet klasifikačních systémů a nomenklatur si vyžádal vytváření různých konverzních nástrojů pro převod mezi hlavními klasifikačními systémy a pro zachycení vztahů mezi termíny v těchto systémech. Modelovány jsou rozsáhlé ontologie a sémantické sítě pro přenos informací mezi různými datovými bázemi, vytvářeny jsou tzv. metatezaury k zachycení a propojení informací z různých heterogeních zdrojů. Nejrozsáhlejším projektem tohoto druhu je v dnešní době UMLS.

3.1. UMLS

Vývoj *Unified Medical Language System*u (UMLS) [7] začal v roce 1986 v *Národní lékařské knihovně* v USA jako "Long-term R&D project". UMLS znalostní zdroje jsou univerzální, to znamená, že nejsou optimalizované pro jednotlivé aplikace. UMLS obsahuje více než 730 000 biomedicínských termínů z více než 50 biomedicínských slovníků. Jedná se o inteligentní automatizovaný systém, který "rozumí" biomedicínským termínům a jejich vztahům a využívá tohoto porozumění ke čtení a organizování informací ze strojově zpracovatelných zdrojů. Jeho cílem je kompenzace terminologických a kódových rozdílů těchto nesourodých systémů a současně i jazykových variací uživatelů. Jedná se o vícejazyčný slovník číselníků MeSH, ICD, CPT, DSM, SNOMED a další) na velkokapacitním médiu, což umožňuje převod kódovaných termínů mezi různými klasifikačními systémy.

UMLS se skládá ze tří znalostních zdrojů: *Metathesaurus*®, *Semantic Network* a *SPECIALIST Lexicon*. Semantic Network obsahuje informace o sémantických druzích a jejich vztazích. Ve SPECIALIST lexikonu každé slovo nebo termín zaznamenává *syntaktické*, *morfologické* a *ortografické* informace.

UMLS Metathesaurus je rozsáhlá, víceúčelová a vícejazyčná lexikonová databáze, která zahrnuje informace o biomedicínských, zdravotnických a jim příbuzným pojmech, obsahuje jejich různé názvy a vztahy mezi nimi. UMLS Metathesarus vznikl z elektronických verzí mnoha různých tezaurů, klasifikací nebo souborů kódů jako jsou například SNOMED, MeSH, AOD, Read Codes, ICD-10 a dalších. Jeho hlavním cílem je spojení alternativních názvů stejných pojmů a identifikování užitečných vztahů mezi různými pojmy. Pokud různé slovníky používají stejný název pro různé termíny, v Metatezauru se objeví oba významy a ukáže se v něm, který význam je použit v kterém slovníku. Pokud se stejný termín objevuje v různých slovnících v různých hierarchických kontextech, v Metatezauru jsou zachyceny všechny tyto hierarchie. Metatezaurus nepodává jeden konzistentní pohled, ale zachovává mnoho pohledů, které jsou obsaženy ve zdrojových slovnících.

Počítačová aplikace, která poskytuje internetový přístup ke znalostním a příbuzným zdrojům se nazývá *UMLS Knowledge Source Server*. Jeho cílem je zpřístupnění UMLS dat uživatelům. Systémová architektura umožňuje vzdáleným uživatelům poslat dotaz do Národní lékařské knihovny. UMLS Knowledge Source Server můžeme nalézt na www stránce http://umlsks.nlm.nih.gov/. Po přihlášení se uživatel dostane na stránky UMLS Knowledge Source Serveru. Zde si nejprve zvolíme verzi, se kterou budeme dále pracovat. Nejaktuálnější verze je 2005AA. Poté vložíme hledaný termín. Objeví se nám identifikační číslo termínu, sémantický druh, definice a synonyma. Jak už bylo zmíněno výše, pro jeden výraz nebo termín existuje v medicíně mnoho synonym. UMLS Knowledge Source Server nám ukáže, ve kterých všech klasifikačních systémech se námi zadaný termín nachází. Zpřístupněny jsou i informace o obdobných termínech, užších případně širších termínech, sémantických vztazích s jinými termíny a další podrobné informace.

Pro moji práci z hlediska první analýzy využitelnosti těchto klasifikačních systémů pro potřeby popsání klinického obsahu některých systémů používaných ve zdravotnictví v České republice je nejdůležitější zjistit a vyhledat, zda se daný termín nachází v klasifikačním systému SNOMED CT a zjistit jeho identifikační číslo v tomto systému. Toto a případně identifikátory v dalších systémech lze později využít při modelování tzv. *archetypů* – základních stavebních kamenů elektronických zdravotních záznamů.

3.2. SNOMED CT

SNOMED Clinical Terms (SNOMED CT) [8] vznikl spojením dvou terminologií: *SNOMED RT* a *Clinical Terms Version 3* (Read Codes CTV3). SNOMED RT představuje *Systematized Nomenclature of Medicine Reference Terminology*, kterou vytvořila *College of American Pathologists*. Slouží jako společná referenční terminologie pro shromažďování a získávání zdravotnických dat zaznamenaných organizacemi nebo jednotlivci. *Clinical Terms Version 3* vznikl v *United Kingdom's National Health Service* v roce 1980 jako mechanismus pro ukládání strukturovaných informací o primární péči ve Velké Británii.

V roce 1999 se tyto dvě terminologie spojily a vznikl tak SNOMED CT, což je vysoce komplexní terminologie. Na jejím vytváření se podílí kolem 50 lékařů, sester, asistentů, lékárníků, informatiků a dalších zdravotnických odborníků z USA a Velké Británie. Byly vytvořeny speciální terminologické skupiny pro specifické terminologické oblasti jako je například ošetřovatelství nebo farmacie. SNOMED CT zahrnuje 364 400 zdravotnických termínů, 984 000 anglických popisů a synonym a 1 450 000 sémantických vztahů.

Mezi oblasti SNOMED CT patří *finding*, *disease*, *procedure and intervention*, *observable entity*, *body structure*, *organism*, *substance*, *pharmaceutical/biological product*, *specimen*, *physical object*, *physical force*, *events*, *environments and geographical locations*, *social context*, *contex-dependent categories*, *staging and scales*, *attribute* a *qaulifier value*. V současné době existuje americká, britská, španělská a německá verze SNOMED CT.

4. Vlastní výzkum

4.1. Využití klasifikačních systémů pro sdílenou zdravotní péči

Mapování terminologie uváděné v aplikacích elektronického zdravotního záznamu na mezinárodně používané terminologické slovníky, tezaury, ontologie a klasifikace je základem pro interoperabilitu heterogenních systémů elektronického zdravotního záznamu. K zajištění interoperability však nestačí pouhé porozumění si na úrovni terminologických výrazů. Dalším předpokladem pro úspěšné sdílení dat mezi různými aplikacemi zdravotního záznamu je harmonizace klinického obsahu. Tato harmonizace nemusí být úplně stoprocentní, pak je ale možné sdílet pouze data, která jsou mezi aplikacemi společná. Interoperabilitu usnadní, pokud si odpovídají tzv. referenční informační modely jednotlivých aplikací zdravotních záznamů. Samozřejmě se nabízejí možnosti vzájemného mapování mezi těmito modely, což je však těžké vzhledem k odlišnému přístupu jednotlivých modelů.

Například *Referenční informační model HL7* (HL7 RIM) [9] představuje model uzavřeného světa definovaného pomocí tříd, jejich atributů a vztahů mezi třídami. Pro další použití v konkrétní oblasti se od tohoto modelu odvozuje takzvaný *Doménový informační model* (D-MIM). Abychom se od takovéhoto modelu dostali ke zprávám nesoucím informace o zdravotním záznamu pacienta, použijeme tzv. *Refined Message Information Model* (R-MIM), který je podmnožinou D-MIM použitou pro vyjádření informačního obsahu jedné nebo více abstraktních struktur zpráv nazývaných též *Hierarchické popisy zpráv*.

Jiným příkladem je CEN TC 251, který definuje v evropském předběžném standardu ENV 13606 (Sdělování elektronických zdravotních záznamů, 4. část - zprávy pro výměnu informací) obsah elektronického zdravotního záznamu pomocí poměrně hrubého modelu specifikujícího 4 základní složky: *Folder* – popisující větší sekce záznamu daného subjektu, *Composition* – reprezentující jeden identifikovatelný příspěvek ke zdravotnímu záznamu daného subjektu, *Headed Section* – obsahující množiny údajů na jemnější úrovni než Composition a *Cluster* – identifikující skupiny údajů, které by měly zůstat seskupeny, hrozí-li ztráta kontextu.

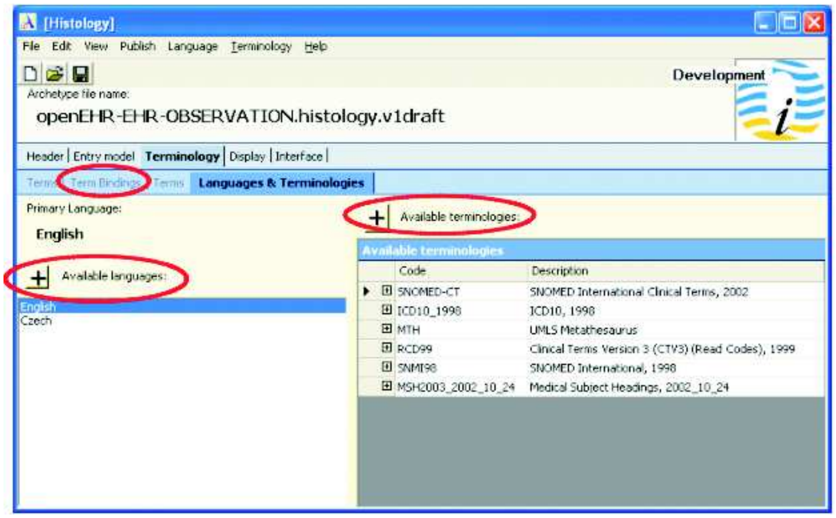
Zcela jiný přístup používá asociace NEMA (*National Electrical Manufacturers Association*) při specifikaci DICOM SR (*DICOM Structured Reporting*), ve které dochází k rozšíření specifikace pro generování, prezentaci, výměnu a archivaci medicínských snímků DICOM na modelování celého zdravotního záznamu pacienta. Hlavní ideou zde je použít existující infrastrukturu DICOM pro výměnu strukturovaných zpráv, které představují hierarchický strom dokumentu s typovanými koncovými uzly. Sémantika jednotlivých uzlů je popsána kódovacími systémy jako např. ICD-10 či SNOMED.

Referenční model *Synapses Object Model* (SynOM) vytvořený v rámci projektu *Synapses*, resp. *SynEx* (*Synergy on the Extranet*) [10] je velmi podobný modelu definovanému v CEN ENV 13606. Jako typy sbíraných hodnot jsou zde využity tzv. *archetypy* - definice strukturovaně sbíraných údajů v určité doméně obsahující specifikovaná omezení zajišťující integritu celkového záznamu. Projekt dále pod záštitou neziskové *openEHR Foundation* pokračoval a definoval tzv. *Good European Health Record* (GEHR) [11]. V projektu odborníci specifikují požadavky elektronického zdravotního záznamu s hlavním cílem podpořit možnosti integrace a spolupráce heterogenních EHR aplikací. Za tímto účelem vznikl formální model specifikující GEHR architekturu (*GEHR Object Model*, GOM) a znalostní model specifikující klinickou strukturu záznamu pomocí archetypů. Výstupy projektu openEHR lze v dnešní době považovat za významnou konkurenci standardům orientovaným na implementační aspekty EHR systémů.

4.2. Mapování terminologie a tvorba archetypů

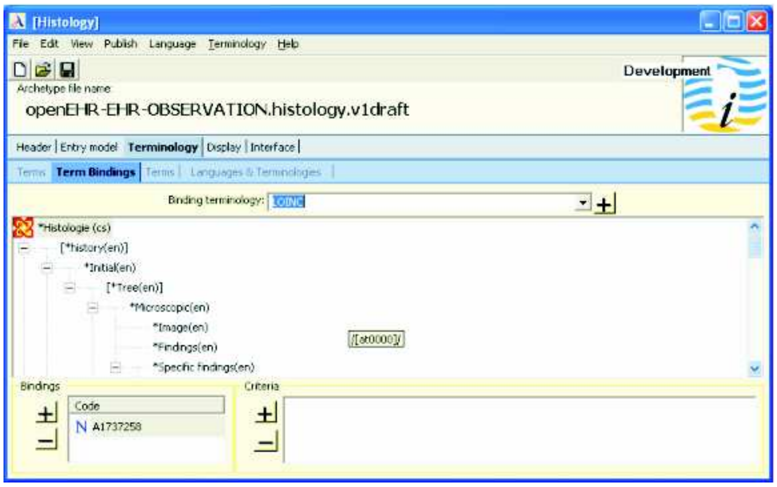
Aby informatici mohli mapování na terminologie do budoucna využít, je vhodné na správnou terminologii myslet již od začátku, tj. jak při navrhování archetypů tak při tvorbě ostatních základních elementů v jiných typech modelů architektury zdravotních záznamů.

Jako příklad, jak správnou terminologii odkazovat již při vytváření archetypů, může fungovat editor od firmy *Ocean Informatics* [12] zobrazený na obrázku číslo 1.



Obrázek 1: Práce s terminologií v editoru archetypů.

Je možné přidat libovolný počet jazyků, ve kterých daný termín popíšeme. Zároveň je možné zvolit z dostupných terminologií ty, které použijeme k tomu, abychom definovali správný význam jednotlivých termínů. Na dalších záložkách v tomto editoru definujeme jednotlivé termíny a na záložce Term bindings provedeme příslušné mapování našich termínů na termíny v terminologických slovnících tak, jak je zobrazeno na obrázku číslo 2.



Obrázek 2: Mapování použité terminologie na standardní kódovací systémy.

4.3. Standardizace klinického obsahu

Analýzu vhodnosti a využitelnosti jednotlivých terminologických slovníků jsem započala mapováním klinického obsahu tzv. *Minimálního datového modelu kardiologického pacienta* (MDMKP) [13] na různé terminologické klasifikační systémy. MDMKP je souborem přibližně 150 atributů, jejich vzájemných vztahů, integritních omezení a jednotek. Na těchto atributech se shodli přední odborníci v oblasti české kardiologie jako na základních údajích nutných při vyšetření kardiologického pacienta.

Při analýze jsem zjistila, že přibližně 85 % atributů MDMKP je obsaženo alespoň v nějakém klasifikačním systému. Většina z nich (přes 50 %) je obsažena v systému SNOMED CT. Atributy z pohledu možnosti jejich mapování na standardní kódovací systémy lze klasifikovat následujícím způsobem:

- Bezproblémové atributy - tj. atributy, které lze mapovat přímým způsobem, tak, že u nich existuje právě jedna možností mapování, případně existují pouze synonyma se zcela stejným významem a tedy i klasifikačním kódem (např.: *křestní jméno pacienta*, *současný kuřák*, *hybnost*, *výška pacienta*).
- Částečně problematické atributy - tj. atributy, které lze mapovat tak, že u nich existuje právě několik možností mapování na různá synonyma, která se ale lehce liší významem a tedy zpravidla i klasifikačním kódem (např.: *ischemická cévní mozková příhoda*, *angína pectoris*, *hypertenze*, *městnavé srdeční selhání*).
- Atributy s příliš malou granularitou, tj. atributy popisující určitou vlastnost na příliš obecné úrovni tak, že klasifikační systémy obsahují pouze termíny užšího významu (např. *email* v MDMKP vs. *email do zaměstnání* / *email domů* / *email lékaře* atd. v klasifikačních systémech).
- Atributy s příliš velkou granularitou, tj. atributy popisující určitou vlastnost na úzké úrovni tak, že klasifikační systémy obsahují pouze termíny obecnějšího významu (např. *šelest nad AO ústím*, *souměrný tep karotid*, atd.).
- Atributy, které se v klasifikačních systémech dohledat nedaří, např. *rodné číslo*, *dyslipidemie*, atd.

K obdobnému závěru jsem dospěla při analýze možností standardizace atributů *Datového standardu Ministerstva zdravotnictví České republiky* (DASTA) [14]. Strukturované atributy v tomto standardu se však ve velké míře omezují na administrativní a laboratorní údaje. Při mapování administrativních údajů byly výsledky obdobné jako při mapování administrativních údajů v MDMKP. Laboratorní údaje jsou v tomto standardu velmi podrobně specifikované pomocí tzv. Národního číselníku laboratorních položek [15], na jehož podrobnější analýze teprve pracuji.

V neposlední řadě se snažím o mapování atributů vybraných klinických modulů komerčních nemocničních informačních systémů. Jako příklad uvedu výsledky mapování specializovaného EKG modulu v klinickém informačním systému *WinMedicalc*. Vzhledem k velké specializovanosti tohoto modulu se podařilo namapovat přibližně 60 % atributů na různé klasifikační systémy. Převládající klasifikační problémy souvisí v tomto případě s příliš velkou granularitou atributů v tomto modelu (*ejekční frakce 1*, *ejekční frakce 2*, *septum levé komory*).

Řešení problémů při mapování je zpravidla nutno provádět v úzké spolupráci s odborníky z řad lékařů. Často je třeba provést volbu vhodného synonyma nahrazující určitý odborný termín. Toto je však třeba provádět s nejvyšší opatrností tak, aby nedošlo ke ztrátě informace případně jejímu zkreslení. V případě, že toto nelze bez ztráty informace, provést, je lepším řešením popsat určitý nekódovatelný termín pomocí skupiny několika kódovatelných termínů, případně též se zachycením vzájemných sémantických vztahů. Není-li ani toto možné, lze polemizovat s příslušnými odborníky, zda by tyto standardně "nepopsatelné" termíny (atributy) nebylo možné nahradit jinými ekvivalentními a standardnějšími. Ve speciálních případech je možné vyvinout aktivitu za účelem přidání určitého termínu do připravované nové verze určitého kódovacího systému. V případě, že není možné použít žádnou z výše uvedených možností řešení problémů při mapování, je třeba smířit se s tím, že mapování nebude nikdy 100%. Nedostatečné mapování pak ale v praxi limituje možnosti interoperability s jinými systémy používanými k různým účelům ve zdravotnictví. Omezená interoperabilita je však často nevyhnutelná již ze samotného jádra problému, například při neúplné harmonizaci klinického obsahu heterogenních systémů elektronického zdravotního záznamu.

5. Souhrn a závěr

Svojí prací se snažím o ověření praktické použitelnosti mezinárodně používaných terminologických slovníků, tezaurů, ontologií a klasifikací a to konkrétně tak, že studuji atributy Minimálního datového modelu kardiologického pacienta, Datového standardu Ministerstva zdravotnictví České republiky a některých vybraných modulů komerčních nemocničních informačních systémů, které dohledávám primárně v klasifikaci SNOMED CT, případně v dalších. SNOMED CT je používán v HL7 verzi 3, a proto se snažím v první řadě mapovat na tento klasifikační systém. V případě neexistence termínu zkouším ostatní dostupné terminologie. Při této práci využívám UMLS Metatezaurus.

Při mapování čelím několika problémům – nejednoznačnosti při mapování a nemožnosti provést mapování z důvodu neexistenci odpovídajícího termínu v klasifikačních systémech. Velkým problémem při využití nomenklatur a metatezaurů ve zdravotnictví v České republice zůstává neexistence českých terminologických systémů či jejich vhodných českých překladů.

I přes problémy, které při využití mezinárodních nomenklatur a metatezaurů ve zdravotnictví v České republice přetrvávají, je jejich využití prvním nezbytným krokem k umožnění interoperability heterogenních systémů zdravotních záznamů. Dostatečná interoperabilita těchto systémů je základem pro sdílenou zdravotní péči, která vede k efektivitě ve zdravotnictví, finančním úsporám i snížení zátěže pacientů, a proto se ve své práci snažím analyzovat, jak mezinárodních klasifikačních systémů využít co nejlépe pro potřeby českého zdravotnictví.

Literatura

[1] World Health Organization©, "Internatinal Classification of Diseases", homepage on the Internet, 2005, available from: http://www.who.int/classifications/icd/en/.

[2] SNOMED International®, "Systematized Nomenclature of Medicine", homepage on the Internet, 2004, available from: http://www.snomed.org/.

[3] National Library of Medicine, "Medical Subject Headings", homepage on the Internet, available from: http://www.nlm.nih.gov/mesh/MBrowser.html.

[4] Regenstrief Institute, Inc., "Logical Observation Identifiers Names and Codes – LOINC®", homepage on the Internet, available from: http://www.regenstrief.org/loinc/.

[5] World Health Organization©, "Internatinal Classification of Diseases for Oncology", 1990, homepage on the Internet, 2005, available from: http://www.cog.ufl.edu/publ/apps/icdo/.

[6] Woxbridge Solutions Ltd©, "General Practice Notebook – a UK Medical Encyclopaedia on the World Wide Web", 2005, http://www.gpnotebook.co.uk/simplepage.cfm?ID=1134166031.

[7] United States National Library of Medicine, National Institute of Health, "Unified Medical Language System", homepage on the Internet, available from: http://www.nlm.nih.gov/research/umls/.

[8] SNOMED International®, "Systematized Nomenclature of Medicine – Clinical Terms", homepage on the Internet, available from: http://www.snomed.org/snomedct/.

[9] Health Level Seven, Inc., "HL7 Version 3 Standards", 2005, homepage on the Internet, available from: http://www.hl7.cz/.

[10] Jung B., Grimson J., "Synapses/SynEx goes XML", *Studies in Health Technology and Informatics*, Vol. 68, 1999, pp. 906-911.

[11] Centre for Health Informatics & Multiprofessional Education (CHIME), "The Good European Health Record", available from: http://www.chime.ucl.ac.uk/work-areas/ehrs/GEHR/.

[12] Miro International Pty Ltd.© , "Ocean Informatics", 2000-2004, homepage on the Internet, available from: http://oceaninformatics.biz/CMS/index.php.

[13] Tomečková M., "Minimální datový model kardiologického pacienta – výběr dat", In: *Cor et Vasa*, Vol. 44, No. 4 Suppl., ISSN: 0010-8650, 2002, p. 123.

[14] Lipka J., Mukenšnábl Z., Horáček F., Bureš V., "Současný komunikační standard českého zdravotnictví DASTA", In: Zvárová J., Přečková P. (eds.): *Informační technologie v péči o zdraví*, EuroMISE s.r.o., Praha, 2004, pp. 52-59.

[15] Ministerstvo zdravotnictví České republiky, "Datový standard MZ ČR, Národní číselník laboratorních položek MZ ČR a Národní zdravotnický informační systém", 2004, homepage on the Internet, available from: http://www.mzcr.cz/index.php?kategorie=31.

Modelling of Piezoelectric Materials

Post-Graduate Student:
ING. PETR RÁLEK

Katedra modelování procesů
TU Liberec
Hálkova 6
461 17 Liberec

Czech Republic

petr.ralek@vslib.cz

Supervisor:
DOC. DR. ING. JIŘÍ MARYŠKA, CSC.

Katedra modelování procesů
TU Liberec
Hálkova 6
461 17 Liberec

Czech Republic

jiri.maryska@vslib.cz

Field of Study:
natural engineering
Classification:

Abstract

In the paper we introduce the application of the mathematical model of piezoelectric resonator. The finite element (FEM) model of the piezoelectric resonator is based on the physical description of the piezoelectric materials. Discretization of the problem then leads to a large sparse linear algebraic system, which defines the generalized eigenvalue problem. Resonance frequencies are subsequently found by solving this algebraic problem. Depending on the discretization parameters, this problem may become large, which may complicate application of standard techniques known from the literature. Typically, we are not interested in all eigenvalues (resonance frequencies). For determining of several of them it seems therefore appropriate to consider iterative methods.

The model was tested on the problem of thickness-shear vibration of plan-parallel quartz resonator. The results are given in the article.

1. Physical description

I briefly sketch the physical properties of the piezoelectric materials. For more detailed description (including more references), see e.g. [6].

A crystal made of piezoelectric material represents a structure in which the deformation and electric field depend on each other. A deformation (impaction) of the crystal induces electric charge on the crystal's surface. On the other hand, subjecting a crystal to electric field causes its deformation. In linear theory of piezoelectrocity, derived by Tiersten in [8], this process is described by two constitutive equations - the **generalized Hook's law** (1) and the **equation of the direct piezoelectric effect** (2),

$$\mathrm{T_{ij}} = c_{ijkl}\,\mathrm{S_{kl}} - d_{kij}\,\mathrm{E_k}, \qquad i,j = 1,2,3, \tag{1}$$

$$\mathrm{D}_k = d_{kij}\,\mathrm{S_{ij}} + \varepsilon_{kj}\,\mathrm{E_j}, \qquad k = 1,2,3. \tag{2}$$

Here, as in other similar terms troughout the thesis, we use the convention known as the Einstein's additive rule ($a_{ij}b_j = \sum_{j=1}^{3} a_{ij}b_j$. The Hook's law (1) describes dependence between the symmetric **stress tensor T**, the symmetric **strain tensor S** and the **vector of intensity of electric field E**,

$$\mathrm{S}_{ij} = \frac{1}{2}\left[\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i}\right], \qquad i,j = 1,2,3, \quad \mathrm{E_k} = -\frac{\partial \tilde{\varphi}}{\partial x_k}, \qquad k = 1,2,3,$$

where $\tilde{\mathbf{u}} = (\tilde{u}_1, \tilde{u}_2, \tilde{u}_3)^{\mathrm{T}}$ is the **displacement vector** and $\tilde{\varphi}$ is the **electric potential**. The equation of the direct piezoelectric effect (2) describes the dependence between the **vector of electric displacement** $\mathbf{D}$, the strain and the intensity of electric field. Quantities c_{ijkl}, d_{kij} and ε_{ij} represent symmetric material tensors, playing role of the material constants. Additional, tensors c_{ijkl} and ε_{ij} are positive definite.

1.1. Oscillation of the piezoelectric continuum

Consider resonator made of piezoelectric material with density ϱ, characterized by material tensors. We denote the volume of the resonator as Ω and its boundary as Γ. Behavior of the piezoelectric continuum is governed, in some time range $(0, \mathrm{T})$, by two differential equations: Newton's law of motion (3)and the quasistatic approximation of Maxwell's equation (4) (see, e.g., [4]),

$$\varrho \frac{\partial^2 \tilde{u}_i}{\partial t^2} = \frac{\partial \mathrm{T_{ij}}}{\partial x_j} \qquad i = 1,2,3, \qquad x \in \Omega, \quad \mathrm{t} \in (0, \mathrm{T}), \tag{3}$$

$$\nabla . \, \mathbf{D} = \frac{\partial \mathrm{D_j}}{\partial x_j} = 0. \tag{4}$$

Replacement of **T**, resp. **D** in (3) and (4) with the expressions (1), resp. (2), gives

$$\varrho \frac{\partial^2 \tilde{u}_i}{\partial t^2} = \frac{\partial}{\partial x_j} \left(c_{ijkl} \, \frac{1}{2} \left[\frac{\partial \tilde{u}_k}{\partial x_l} + \frac{\partial \tilde{u}_l}{\partial x_k} \right] + d_{kij} \, \frac{\partial \tilde{\varphi}}{\partial x_k} \right) \quad i = 1,2,3, \tag{5}$$

$$0 = \frac{\partial}{\partial x_k} \left(d_{kij} \, \frac{1}{2} \left[\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i} \right] - \varepsilon_{kj} \, \frac{\partial \tilde{\varphi}}{\partial x_j} \right). \tag{6}$$

Initial conditions, Dirichlet boundary conditions and Neumann boundary conditions are added:

$$\begin{aligned}
\tilde{u}_i(.,0) &= u_i, \quad x \in \Omega, \\
\tilde{u}_i &= 0, \qquad i = 1,2,3, \quad x \in \Gamma_u, \\
\mathrm{T_{ij}} n_j &= f_i, \qquad i = 1,2,3, \quad x \in \Gamma_f, \\
\tilde{\varphi}(.,0) &= \varphi, \\
\tilde{\varphi} &= \varphi_D, \quad x \in \Gamma_\varphi \\
\mathrm{D_k} n_k &= q, \quad x \in \Gamma_q,
\end{aligned} \tag{7}$$

where

$$\Gamma_u \cup \Gamma_f = \Gamma, \ \Gamma_u \cap \Gamma_f = \emptyset, \ \Gamma_\varphi \cup \Gamma_q = \Gamma, \ \Gamma_\varphi \cap \Gamma_q = \emptyset.$$

Right-hand side f_i represents mechanical excitation by external mechanical forces, q denotes electrical excitation by imposing surface charge (in the case of free oscillations, they are both zero). Equations (5)-(6) define the problem of harmonic oscillation of the piezoelectric continuum under given conditions (7).

We will discretize the problem using FEM. Its basic formulation was published by Allik back in 1970 [1], but the rapid progress in FEM modelling in piezoeletricity came in the last ten years.

2. Weak formulation

Discretization of the problem (5)-(7) and the use of the finite element method is based on so called *weak formulation*. We briefly scetch the function spaces used in our weak formulation. We deal with the weak formulation derived in [7], chapters 28-35. For more details we recommend the reader to this book. We consider bounded domain Ω with Lipschitzian boundary Γ. Let $L_2(\Omega)$ be the Lebesgeue space of functions square integrable in Ω. *Sobolev space* $W_2^{(1)}(\Omega)$ is made of functions from $L_2(\Omega)$, which have generalized derivatives square integrable in Ω. To express values of function $u \in W_2^{(1)}(\Omega)$ on the boundary Γ, the *trace* of function u is established (see [7]; for function from $C^{(\infty)}(\overline{\Omega})$, its trace is determined by its values on the boundary). We establish

$$V(\Omega) = \{ v | v \in W_2^{(1)}(\Omega), \quad v|_{\Gamma_1} = 0 \text{ in the sence of traces} \},$$

the subspace of $W_2^{(1)}(\Omega)$, made of functions, which traces fulfil the homogenous boundary conditions.

We derive the weak formulation in the standard way ([7], chapter 31). We multiply the equations (5) with testing functions $w_i \in V(\Omega)$, summarize and integrate them over Ω. As well, we multiply the equation (6) with testing function $\phi \in V(\Omega)$ and integrate it over Ω. Using Green formula, symmetry of material tensors and the boundary conditions, we obtain the integral equalities (boundary integrals are denoted with sharp brackets)

$$\left(\varrho \frac{\partial^2 \tilde{u}_i}{\partial t^2}, w_i\right)_\Omega + \left(c_{ijkl}\, \mathrm{S}_{kl}, \mathrm{R}_{ij}\right)_\Omega + \left(d_{kij}\, \frac{\partial \tilde{\varphi}}{\partial x_k}, \mathrm{R}_{ij}\right)_\Omega = \left\langle f_i, w_i \right\rangle_{\Gamma_f}, \tag{8}$$

$$\left(d_{jik}\, \mathrm{S}_{ik}, \frac{\partial \phi}{\partial x_j}\right)_\Omega - \left(\varepsilon_{ji}\, \frac{\partial \tilde{\varphi}}{\partial x_i}, \frac{\partial \phi}{\partial x_j}\right)_\Omega = \left\langle q, \phi \right\rangle_{\Gamma_q}. \tag{9}$$

Weak solution: Let

$$\tilde{\mathbf{u}}_D \in ([W_2^{(1)}(\Omega)]^3, C^{(2)}(0,\mathrm{T})),\ \tilde{\varphi}_D \in (W_2^{(1)}(\Omega), AC(0,\mathrm{T}))$$

satisfy the Dirichlet boundary conditions (in the weak sence). Further, let

$$\tilde{\mathbf{u}}_0 \in ([W_2^{(1)}(\Omega)]^3, C^{(2)}(0,\mathrm{T})),\ \varphi_0 \in (W_2^{(1)}(\Omega), AC(0,\mathrm{T}))$$

be functions, for which equalities (8) and (9) are observed for all choices of testing functions

$$\mathbf{w} = (w_1, w_2, w_3) \in [V(\Omega)]^3,\ \phi \in V(\Omega).$$

Then we define the *weak solution* of the problem (5)-(7) as

$$\tilde{\mathbf{u}} = \tilde{\mathbf{u}}_D + \tilde{\mathbf{u}}_0, \quad \tilde{\varphi} = \tilde{\varphi}_D + \tilde{\varphi}_0.$$

Weak solution, on the contrary to the classical solution, does not necesarilly have continuous spatial derivatives of the 2nd order. The weak solution has generalized spatial derivatives and statisfies the integral identities (8), (9).

3. Discretization of the problem

We discretize the problem in space variables, using tetrahedron elements with linear base functions (Fig. 2) The system of ordinary differential equations for values of displacement and potential in the nodes of division results. It has block structure,

$$\mathrm{M}\ddot{\mathrm{U}} + \mathrm{KU} + \mathrm{P}^{\mathrm{T}}\Phi = \mathrm{F}, \tag{10}$$

$$\mathrm{PU} - \mathrm{E}\Phi = \mathrm{Q}. \tag{11}$$

After introduction of Dirichlet boundary conditions (see Fig. 1), sub-matrices M, K and E are symmetric and positive definite. For detailed description of discretization process see [1] and [5] or [6].

The core of the behavior of the oscillating piezoelectric continuum lies in its free oscillation. Free oscillations (and computed eigenfrequencies) tell, when the system under external excitation can get to the resonance. For the free harmonic oscillation, the system (10) can be transformed to

$$\begin{pmatrix} \mathrm{K} - \omega^2\mathrm{M} & \mathrm{P}^{\mathrm{T}} \\ \mathrm{P} & -\mathrm{E} \end{pmatrix} \begin{pmatrix} \mathrm{U} \\ \Phi \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \tag{12}$$

where ω is the frequency of oscillation. Eigenfrequencies can be computed by solving the generalized eigenvalue problem

$$\mathrm{AX} = \lambda \mathrm{BX} \tag{13}$$

Table 1: Comparison of measured and computed resonance frequencies.

Resonator Sample	R (mm)	r (mm)	h (mm)	Measured res. frequency (kHz)	Computed res. frequency (kHz)
1	7	3.5	0.3355	5000.200	5 025
2	3.975	2.5	0.168	10000.125	10104
3	3.475	1.5	0.0833	19990.700	20100

with

$$\mathrm{A} = \begin{pmatrix} \mathrm{K} & \mathrm{P}^{\mathrm{T}} \\ \mathrm{P} & -\mathrm{E} \end{pmatrix},\ \mathrm{B} = \begin{pmatrix} \mathrm{M} & 0 \\ 0 & 0 \end{pmatrix},\ \lambda = \omega^2,$$

where A is symmetric and B is symmetric and positive semi-definite matrix. Computed eigenvectors (resp. part U) describe the modes of oscillations. For solving the generalized eigenvalue problem (13), we use implicitly restarted Arnoldi method implemented in Arpack library [9] (in Fortran language). Inner steps in the process use algorithm SYMMLQ [11] for solving the symmetric indefinite linear systems. It is suitable for solving partial eigenvalue problem with possibility of the shift and it allows to deal with the sparsity of the matrices. The method solves the partial eigenvalue problem (computes several eigenvalues with high precision). Using of the shift enables to obtain the eigenvalues from desired part of the spectrum with better accuracy, than in the approach mentioned bellow.

Other possibility is to use the static condensation, i.e. to transform the problem (13) to the positive definite eigenvalue problem

$$\mathrm{K}^\star\mathrm{U} = \lambda\mathrm{M}\mathrm{U},\ \mathrm{K}^\star = \mathrm{K} - \mathrm{P}^{\mathrm{T}}\mathrm{E}^{-1}\mathrm{P}. \tag{14}$$

This approach was used in [2]. It has many disadvantages, e.g. the loss of sparsity of matrix $\mathrm{K}^\star$ or necessity of computation of the matrix $\mathrm{P}^{\mathrm{T}}\mathrm{E}^{-1}\mathrm{P}$. For solving eigenvalue problem (14), the algorithm based on generalized Schur decomposition, implemented in Lapack library [10], was used. It solves the complete eigenvalue problem and therefore is suitable only for problem of low dimensions (coarse meshes). This algorithm is (on the same sizes of problem) about 10 times slower. We exposed this method and show here results obtained by the implicitly restarted Arnoldi method, which allows us to deal with the linear system without necessity of any transformations.

4. Practical problem - Oscillation of Plan-parallel Quartz Resonator

The model was applied on the problem of oscillation of the plan parallel quartz resonator (Fig. 2) in shear vibration mode in one direction. The dimensional parameters for three different shapes of the resonator (oscillating at three different resonance frequencies) are listed in the (Tab. 1). The table includes comparison between computed results and measurement (these resonators are manufactured and their behavior is well known).

The (Fig. 1) describes parts of the computation process.

The preprocessing part consists of building the geometry (according to the engineering assignments) and mesh of the resonator, see (Fig. 2). We use the GMSH [12] code. The processing part computes the global matrices and the consecutive eigenvalue problem (using text file with parameters - accuracy, number of computed eigenvalues, shift, etc...). It gives several output files, which are used in the postprocessing. Computed eigenvalues and eigenvectors define the oscillation modes, which are sorted according their *electromechanical coupling coefficients*. The electromechanical coupling coefficient k is defined [3]

$$k^2 = \frac{E_m^2}{E_{st}E_d},$$

where

$$E_m = \frac{1}{2}\left(\mathrm{U}^t\mathrm{P}\Phi\right)$$

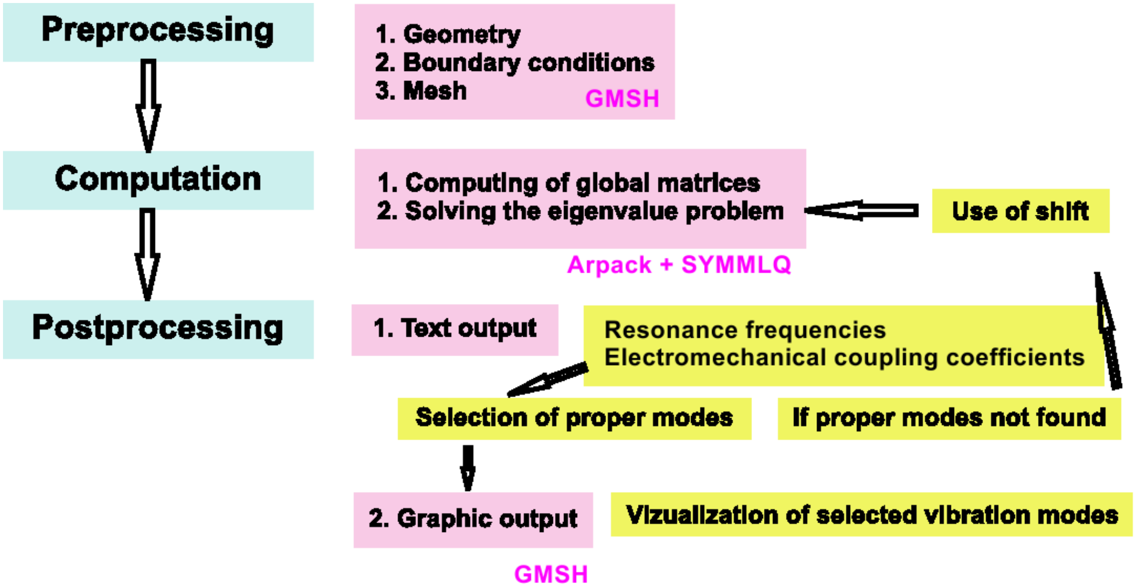


Figure 1: Description of the parts of computation

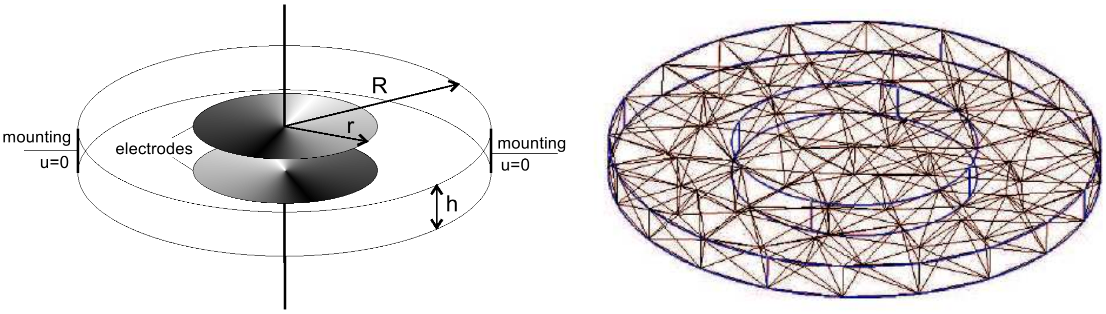


Figure 2: Geometry and discretization of plan-parallel resonator

is the mutual energy,

$$E_{st} = \frac{1}{2}\left(\mathrm{U}^t\mathrm{K}\mathrm{U}\right)$$

is the elastic energy and

$$E_d = \frac{1}{2}\left(\Phi^t\mathrm{E}\Phi\right)$$

is the dielectric energy. The higher is the value of k, the better is the possibility excitation of the oscillation mode. The (Fig. 3) shows the graph of coefficients k for the part of spectra about 5 MHz and the selection of modes with highest coefficients. Selected modes can be then displayed in GMSH (Fig. 4). As a remark, in the (Fig. 4) is shown the dependance of number of inner iterations of SYMMLQ in the process of solving the eigenvalue problem to the size of the problem. The dependance is roughly linear, which is quite positive. In the same figure, there are shown the reziduas after the computation, which are of order 10^{-13} in magnitude in the worst case.

5. Conclusion

The presented mathematical model gives suitable results for the testing problems. It uses methods of numerical linear algebra for solving the partial generalized eigenvalue problem with possibility of shift mode, which allows to compute eigenfrequencies in the neighborhood of the desired value. The restarted Arnoldi

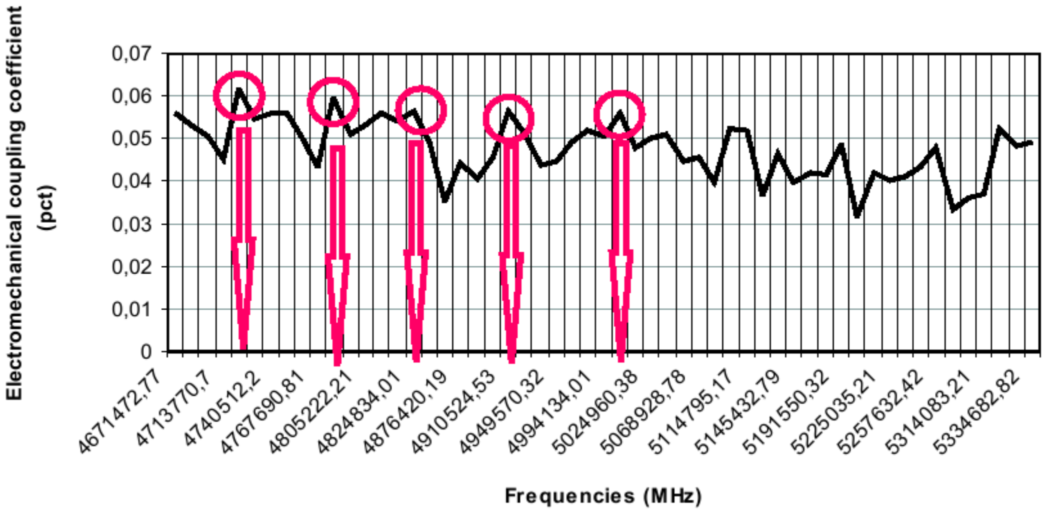


Figure 3: Selection of the dominant modes.

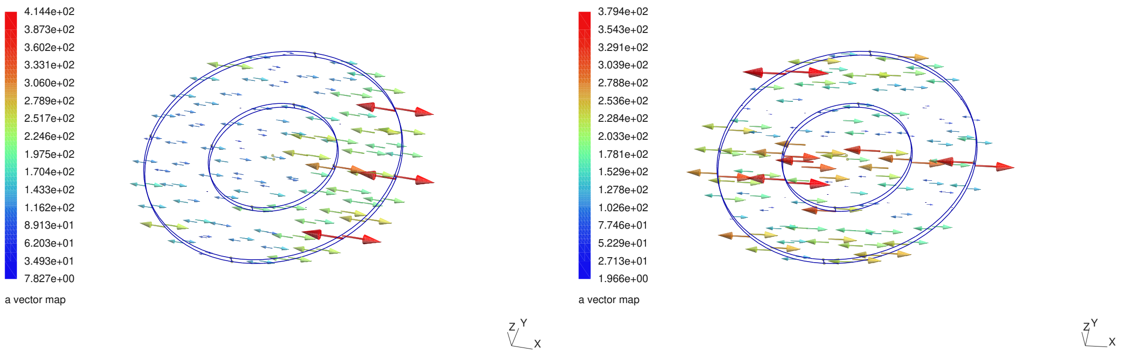


Figure 4: Two examples of oscillation modes

methods looks pretty effective for this problem. Used numerical method brings significant improvement to the method used in [2] and it is suitable for solving larger problems originated by discretization of more complicated shapes of resonators. The difference between calculated and measured results can be caused by several reasons - mainly in the mathematical formulation, it the use of the simply, linear piezoelectric state equations; in the process numerical solution, it is the case of rounding errors during the computation (both in discretization and solving the eigenvalue problem of large dimension).

Nowadays, the next step to do is computation of the graphs of dependance of certain resonance frequency to the geometrical characteristic of the resonator and also the distance of carrier resonance frequency from the spurious frequencies. It still remains as a big task, to improve the postprocessing part of the program for classification of the computed oscillation modes - mainly according to the graphs of amplitudes in several sections of the volume of the resonator.

References

[1] Allik H., Hughes T.J.R., "Finite element method for piezoelectric vibration", *International journal for numerical methods in engineering*, vol. 2, pp. 151-157, 1970.

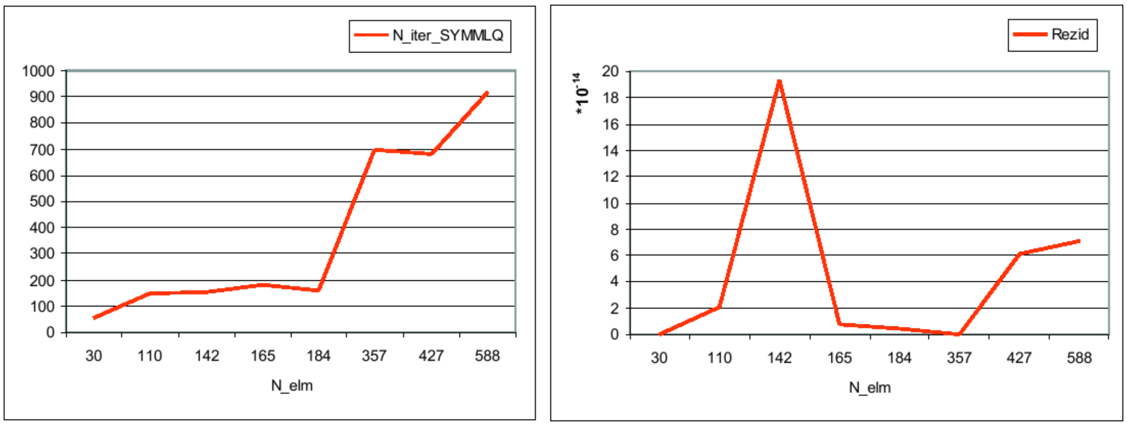


Figure 5: Number of iterations and reziduum.

[2] P. Rálek, J. Maryška, J. Novák: *Modelling of the Resonance Characteristics of the Piezoelectric Resonators - Experimental Experience*, In: Proceedings of ECMS 2003, , pp. , 2003.

[3] Lerch R., "Simulation of Piezoelectric Devices by Two- and Three-Dimensional Finite Elements", *IEEE Trans. on Ultrason., Ferroel. and Frequency Control*, Vol. 37, No. 2 (1990), pp. 233- 247.

[4] Milsom R.F., Elliot D.T., Terry Wood S., Redwood M., "Analysis and Design of Couple Mode Miniature Bar Resonator and Monolothic Filters", *IEEE Trans Son. Ultrason.*, Vol. 30 (1983), pp. 140-155.

[5] Rálek P., "Modelování piezoelektrických jevů", *Diploma thesis, FJFI CVUT*, Prague 2001.

[6] Rálek P., "Modelling of piezoelectric materials", *Proceedings of the iX. PhD. Conference ICS, Academy of Sciences of the Czech Republic*, Prague 2004.

[7] Rektorys K., "Variační metody", *Academia Praha*, 1989.

[8] Tiersten H.F., "Hamilton's principle for linear piezoelectric media", *Proceedings of IEEE 1967*, pp. 1523-1524.

[9] R. Lehoucq, K. Maschhoff, D. Sorensen, Ch. Yang: *www.caam.rice.edu/software/ARPACK/*

[10] *www.netlib.org/lapack/*

[11] C. C. Paige, M. A. Saunders: *http://www.stanford.edu/group/SOL/software/symmlq.html*

[12] Ch. Geuzaine, J.-F. Remacle: *http://www.geuz.org/gmsh/*

Odhadování struktury dat pomocí pravidlových systémů

doktorand:
ING. MARTIN ŘIMNÁČ

Ústav informatiky AV ČR
Pod Vodárenskou věží 2
182 07 Praha 8
rimnacm@cs.cas.cz

školitel:
ING. JÚLIUS ŠTULLER, CSC

Ústav informatiky AV ČR
Pod Vodárenskou věží 2
182 07 Praha 8
stuller@cs.cas.cz

obor studia:
Databázové systémy
číselné označení: II

Práce byla podpořena projektem 1ET100300419 programu Informační společnost "Inteligentní modely, algoritmy, metody a nástroje pro vytváření sémantického webu" a částečně i výzkumným záměrem AV0Z10300504 "Informatika pro informační společnost: Modely, algoritmy, aplikace".

Abstrakt

Metoda odhadování struktury dat spojuje vizi sémantického webu a dnešní webové datové zdroje, které převážně neobsahují žádnou doprovodnou sémantiku prezentovaných informací. Aby bylo možné tyto zdroje použít pokročilými nástroji sémantického webu, je potřeba sémantiku prezentovaných dat alespoň odhadnout. Příspěvek popisuje takovou metodu, ukazuje její použití pro úlohy induktivního logického programování a jmenuje výhody použití pravidlových systémů pro její implementaci.

Jednou z klíčových otázek dnešní doby je zpřístupnění informací nejen lidem, ale i výpočetním prostředkům, které by na dotaz uživatele získaly potřebné informace a provedly by na jejich základě potřebné úkony (integrace z více zdrojů, odvozování informací, zahrnutí předdefinovaných uživatelských profilů nebo preferencí) tak, aby uživatel získal informaci ve zkompletované, přehledné formě.

Možnou odpovědí na tyto otázky je vize sémantického webu [1], tedy rozšíření současných webových zdrojů o dokumenty navíc vhodné ke strojovému zpracování. Tyto dokumenty by vedle samotné informace obsahovaly i její popis - sémantiku, což by umožnilo strojové zacházení s těmito dokumenty. V dnešní době se jeví jako perspektivní formát RDF (Resource Description Framework) [2], popisující realitu pomocí binárních predikátů $vlastnost(objekt, subjekt)$ s návazností na deskripční logiky jako odvozovacího mechanizmu, nebo formát OWL (Web Ontology Language) [3], mapující realitu pomocí vztahů mezi množinami.

Avšak vybrané typy současných webových zdrojů (např. webová rozhraní pro databáze) nemusí být ochuzeny o možnosti sémantického webu, neboť i ony v jistém smyslu obsahují sémantiku, byť v nějaké implicitní formě. Takovou formou může být např. tabulka nebo poloha hodnoty v šabloně designu webové stránky (u XHTML stránek lze data extrahovat pomocí XPath dotazů) apod. Z takové formy je možné odhadnout strukturu dat (např. relační model známý z teorie databází) a z tohoto modelu následně i sémantiku. Informace ze zdrojů pak mohou být extrahovány a uloženy buď do databází nebo do formátů vhodnějších pro sémantický web. Začlenění dat takových dokumentů do portfolia sémantického webu je možné řešit integrací dokumentů sémantického webu.

Příspěvek popisuje jednu takovou metodu [4] odhadu, na jejímž vstupu je tabulka dat s označenými sloupci a výstupem je relační model dat. Metoda byla nejprve implementována pomocí uložených procedur v databázi Postgres [5], které ale většinou představovaly pravidla (Když-Pak). Z tohoto důvodu byla zvolena ještě implementace v pravidlovém systému Clips [6] popsaná v samostatném odstavci.

1. Odhadování struktury dat

Odhadování struktury dat vychází z metod dekomponujících databázový model [7, 8, 9]. Úlohou těchto metod je upravit vstupní model popsaný pomocí množiny funkčních závislostí na nový model tak, aby splňoval další požadavky, např. vyšší normální formu či automatické rozšíření modelu o další vlastnosti, např. modely označované jako multi-level secure [10]. Výstupem těchto metod je model popsaný množinou funkčních závislostí. Aby byl výčet dekompozičních metod kompletní, uveďme ještě metody označované jako vertical a horizontal partitioning [11, 12], sloužící k dekompozici modelu s ohledem na paralerizaci přístupu k datům.

Na rozdíl od výše uvedených přístupů, metoda odhadování struktury dat [4] získává model pouze z dat, vstupem metody je množina tabulka dat a výstupem je model, minimální množina funkčních závislostí platných na množině vstupních dat.

Metoda je primárně vyvíjena jako doplněk současných web-miningových metod [13]. Ty operují především nad metadaty webových stránek a zprostředkovávají o nich souhrnné informace. Navrhovaný doplněk rozšiřuje tyto metody o extrakci samotných prezentovaných dat a podrobuje je analýze a další agregaci.

1.1. Základní vlastnosti funkčních závislostí

V tomto odstavci zopakujeme některé základní vlastnosti známé z teorie relačních databází.

Pokud dva atributy jsou vzájemně funkčně závislé, mají shodnou velikost aktivních domén.

$$A_1 \to A_2 \land A_2 \to A_1 \Rightarrow \|\mathscr{D}_\alpha(A_1)\| = \|\mathscr{D}_\alpha(A_2)\| \tag{1}$$

Množina funkčních závislostí vykazuje transitivitu, tedy:

$$A_1 \to A_2 \land A_2 \to A_3 \Rightarrow A_1 \to A_3 \tag{2}$$

Funkční závislost mezi atributy může existovat pouze v případě, kdy velikost aktivní domény závislého atributu není větší nežli velikost aktivní domény atributu, na němž závisí.

$$A_1 \to A_2 \Rightarrow \|\mathscr{D}_\alpha(A_1)\| \geq \|\mathscr{D}_\alpha(A_2)\| \tag{3}$$

Triviální funkční závislosti jsou ty, které nepopisují vlastnosti modelu, platí nezávisle na něm. Mezi ně patří např.

$$\begin{aligned} A_i &\to A_i \\ A_i &\to \oslash \end{aligned} \tag{4}$$

Komplexní atribut slučuje několik atributů v jeden celek. Pokud (komplexní) atribut H funkčně závisí na (komplexním) atributu G, funkční závislost atributu H na atributu G rozšířeném o libovolný další atribut je triviální.

$$G \to H \Rightarrow G' \to H \quad \forall G' \supset G \tag{5}$$

1.2. Algoritmus odhadu struktury dat

Mějme množinu vstupní tabulku dat (relaci) o n sloupcích a hledejme minimální množinu funkčních závislostí, která daná data popisuje.

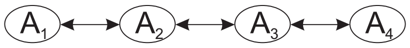
Algoritmus inicializujeme prvním prvkem množiny dat. V této chvíli můžeme hovořit o modelu obsahujícím n^2 funkčních závislostí $A_j \to A_i$, budeme-li uvažovat též komplexní atributy, pak model pokrývá $n!$ (i triviálních) funkčních závislostí.

Výstupem algoritmu je minimální množina funkčních závislostí, `kostra modelu`, reprezentující elementární vazby v modelu. Taková množina neobsahuje žádné triviální funkční závislosti (4, 5) a obsahuje pouze ty funkční závislosti, které tvoří jádro množiny všech netriviálních funkčních závislostí.

Hledání takového jádra vůči jejímu transitivnímu uzávěru je však NP úplná úloha [7, 14] a nemá jedinečné řešení. Proto navržený algoritmus vytvoří v prvním kroku triviálním způsobem kostru modelu a takto vytvořenou kostru v každém kroku aktualizuje, přičemž problém aktualizace je již polynomiálně řešitelný.

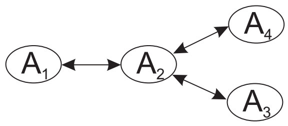
Povšimněme si, že po přidání prvního záznamu každý atribut je extensionálně funkčně závislý na každém jiném atributu (neexistuje žádný další záznam, který by takovou závislost porušoval), tyto závislosti jsou vzájemné. Diskutujme nyní, jaké jsou možné konfigurace kostry modelu a způsoby jejich odvození.

První způsob, `lineární kostra` na obrázku 1, spočívá v náhodném uspořádání atributů a umístěním všech orientovaných hran mezi sousedícími atributy do kostry modelu. Takových uspořádání je faktoriální počet, metoda však na něm dále nezávisí. Možnou nevýhodou je, že takováto kostra modelu obsahuje cykly délky až $2(n-1)$.



Obrázek 1: Kostra v lineární konfiguraci

Tuto potenciální nevýhodu odstraňuje druhý způsob, `star kostra` na obrázku 2, kdy je náhodně vybrán jeden z atributů a kostru modelu tvoří funkční závislosti mezi tímto atributem a ostatními. Je zřejmé, že každý cyklus nabývá buď délky 2 (vzájemná funkční závislost) nebo délky 4. Počet takových modelů je roven počtu atributů, tj. n. Možnou nevýhodou je větší počet změn v kostře při porušení funkční závislosti.



Obrázek 2: Kostra ve "star" konfiguraci

Další možné konfigurace funkčních závislostí, které jsou kostrou, musejí být vytvořeny kombinací těchto dvou přístupů, vlastnosti takových koster se pohybují v rozmezí vlastností obou způsobů.

Přidejme nyní do úložiště další záznam. Předpokládejme, že některé atributy nabývají stejné hodnoty a některé hodnoty jiné. To podle (3) znamená, že bude porušena některá z funkčních závislostí.

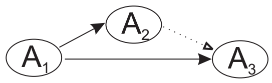
V okamžiku, kdy je do úložiště přidán další záznam, je nutné nejprve aktualizovat kostru modelu. Abychom zachovali daný požadavek "elementárnosti vazeb v kostře", definujme primární kritérium uspořádání atributů s ohledem na (3) tak, že

$$\|\mathscr{D}_\alpha(A_i)\| < \|\mathscr{D}_\alpha(A_j)\| \Rightarrow i < j \tag{6}$$

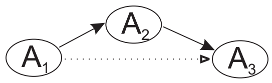
Během aktualizace kostry při změně pořadí atributů může dojít k situaci (7) nebo (8), kdy existuje kratší hrana (podle (7) obrázek 3), která je doplňkem hran tvořících kostru (délku hrany $\delta(A_i, A_j)$ reprezentuje rozdíl pozic v uspořádání atributů, S je množina funkčních závislostí tvořící kostru a $\overline{S}$ je transitivní uzávěr této množiny).

$$(A_1 \to A_3) \in S \wedge (A_1 \to A_2) \in S \wedge (A_2 \to A_3) \in \overline{S} \wedge \delta(A_1, A_3) > \delta(A_2, A_3) \tag{7}$$
$$(A_1 \to A_3) \in S \wedge (A_2 \to A_3) \in S \wedge (A_1 \to A_2) \in \overline{S} \wedge \delta(A_1, A_3) > \delta(A_1, A_2) \tag{8}$$

Jak ukazuje obrázek 4, tato hrana se pak stává hranou tvořící kostru na úkor delší z hran.



Obrázek 3: Kostra $\delta(A_1, A_3) > \delta(A_2, A_3)$

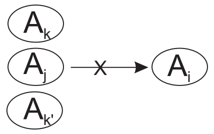


Obrázek 4: Kostra po aktualizaci

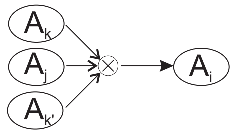
Předpokládejme, že kostra modelu je aktualizována. Otestujme nyní, zda-li některé funkční závislosti nejsou porušeny.

Protože kostra tvoří jádro množiny, pokud je porušena jakákoli funkční závislost v modelu, musí být porušena některá z funkčních závislostí $A_j \to A_i$ tvořících kostru. Díky tomuto faktu není nutné testovat při každém kroku všechny funkční závislosti, ale pouze ty, které tvoří kostru (takových je nejvýše $2n$). V případě, kdy je taková funkční závislost porušena, je nutné navíc testovat i funkční závislosti se stejnou stranou a aktualizovat kostru (najít jiné funkční závislosti s minimální délkou spojující atributy původně spojené přes A_i a A_j).

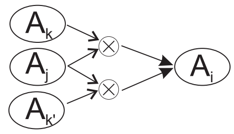
Přidejme další záznamy a předpokládejme, že při testování funkčních závislostí v aktuálním kroku je porušena závislost $A_j \to A_i$, přičemž v minulosti byly porušeny i funkční závislosti $A_k \to A_i$ a $A'_k \to A_i$ a mezi atributy A_k, A'_k a A_i neexistuje žádná funkční závislost. To může vést ke vzniku netriviální závislosti na komplexním atributu $\{A_j, A_k, A'_k\}$, viz obrázek 5. Vložme tedy tento `komplexní atribut` pomocí



Obrázek 5: Porušení $A_j \to A_i$



Obrázek 6: Virtuální atribut



Obrázek 7: Dekompozice

notace `virtuálního atributu` do modelu, virtuální atribut reprezentující kartézský součin je použit, aby nebylo nutné úlohu řešit v prostoru hypergrafů, a otestujme funkční závislost $\{A_j, A_k, A'_k\} \to A_i$. Pokud tato funkční závislost je splněna, virtuální atribut je v modelu ponechán - obrázek 6.

U komplexních atributů arity větší než 2 je navíc pomocí dekompozice komplexního atributu nutné testovat (obrázek 7), zda-li daná závislost není triviální (5). Pokud je, virtuální atribut je dekomponován. Všechny získané netriviální funkční závislosti z této operace jsou přidány do kostry modelu a model je rozšířen i o další neporušené funkční závislosti mezi novými virtuálními atributy a ostatními atributy v modelu.

Jak je patrné, vznik netriviální funkční závislosti na komplexním atributu o m atributech je možný za podmínky, že existuje $m \leq m'$ (alespoň m z celkových m') porušených funkčních závislostí na atributu A_i, který bude na hledaném komplexním atributu závislý a které nemají mezi sebou žádné jiné funkční závislosti. Operace hledání komplexního atributu je nepolynomiálně složitá, je potřeba otestovat celkem až k kombinací (dekompozice atributu), přičemž (při uvažování celočíselného dělení)

$$k = \max_{\forall i < m'} \left(\begin{array}{c} m' \\ i \end{array} \right) = \left(\begin{array}{c} m' \\ m'/2 \end{array} \right) = \frac{m'!}{(m'/2)!^2} \tag{9}$$

Jedinou možností, kdy bude hledání komplexního atributu polynomiálně řešitelné, je díky (3) případ, kdy velikost aktivní domény komplexního atributu roste rychleji než velikost aktivní domény atributu na komplexním atributu závislém.

$$(A \to B),\ A = \{A_0 \ldots A_m\} : \mathscr{D}_\alpha(A) \geq \prod_{i=0}^{m} \mathscr{D}_\alpha(A_i) > \mathscr{D}_\alpha(B) \tag{10}$$

To je ovšem velmi speciální případ, v praxi nastávající zřídkakdy. Příkladem může být tabulka popisující funkční závislost paritního bitu na vzoru. V případě, že je možné vstupní data předzpracovat, je vhodné je uspořádat podle takového kritéria, čímž se proces odhadování modelu urychlí (např. udržovat seznam vhodných trénovacích příkladů pro opakovaný odhad modelu).

Připomeňme, že ostatní operace v metodě jsou vždy polynomiálně složité, tedy hledání netriviálních funkčních závislostí komplexních atributů je jedinou operací, které z celé metody činí nepolynomiálně složitý algoritmus.

Obecně existují dva přístupy ke hledání netriviálních funkčních závislostí nových komplexních atributů. První vychází ze složení komplexního atributu s aritou m a jeho postupné dekompozici na jednodušší komplexní atributy (viz obrázek 6, 7). Naopak druhý přidává další atributy k atributu, na němž byla funkční závislost porušena, tak dlouho, dokud není dosaženo funkční závislosti nebo arity m'. Oba přístupy vedou ke shodnému výsledku, vhodnost použití může být dána heuristikou odvíjející se od hodnoty m'.

Algoritmus 1

```
A .. množina atributů
D .. množina dat
C .. matice pokrytí
S .. kostra modelu
C = {c_ij = 1, i ≠ j, ∀i, j ∈ 1..|A|}
S = {A_i → Aj : |i − j| = 1}
Pro ∀d ∈ D
{
    ulož d do úložiště
    aktualizuj velikost domén a podle změn i pořadí atributů v S a C
    dokud (zmena)
   {
      ∀(i, j, k), i < k < j : (A_i → A_j) ∈ S ∧ (A_k → A_j) ∈ S ∧ C_ik = 1
        S = S − {A_i → A_j} ∪ {A_i → A_k}
      ∀(i, j, k), i < k < j : (A_i → A_j) ∈ S ∧ (A_i → Ak) ∈ S ∧ C_kj = 1
        S = S − {A_i → A_j} ∪ {A_k → A_j}
    }
    testuj ∀(A_i → A_j) ∈ S
    {
      pokud A_i → A_j porušeno
      {
        testuj ∀(A_u → A_v), kde C_iv = 1 ∨ C_uj = 1
        pokud porušeno, C_uv = 0
      }
      S = S − (A_i → Aj)
      rozšiř kostru
      testuj komplexní atributy
    }
}
```

1.3. Vlastnosti modelu

Uspořádejme modely podle počtu všech funkčních závislostí pokrytých modelem. Označme n počet atributů, m počet záznamů, M_k pak model po k-tém záznamu a $|M_k|$ počet funkčních závislostí pokrytých modelem M_k. Podle algoritmu 1 počet hran monotónně klesá, tedy

$$|M_0| = n! \tag{11}$$

$$M_i < M_j \Rightarrow |M_i| > |M_j| \tag{12}$$

$$\forall M_k : M_0 \leq M_k \leq M_m \leq M_\infty \tag{13}$$

Označíme-li M_∞ nejpřesnější (logický) model reality, poslední nerovnost značí, že model vrácený algoritmem může oproti tomuto modelu obsahovat navíc některé funkční závislosti. Tento rozdíl může být způsoben nereprezentativními daty (jedná se o algoritmus strojového učení, kde reprezentativnost dat hraje marginální roli), jednak granularitou hodnot domén jednotlivých atributů (konečný počet záznamů versus nespočitatelné domény atributů v realitě) spojenou s uvažováním extensionálních funkčních závislostí. Poslední možností je možná závislost dat na zdroji (jak samotných hodnot, tak struktury dat). Z těchto důvodů hovoříme o odhadu struktury, nikoli o její rekonstrukci. První a poslední problém lze vyřešit integrací dat z více zdrojů, druhý pak představuje principiální limit metody.

Metoda je určena pro deterministická data neobsahující chybné příklady a předpokládá konzistenci dat v rámci každého zdroje. Pokud tyto podmínky nejsou splněny, mohou být některé funkční závislosti obsažené v logickém modelu z kostry vyjmuty (existují, byť chybné, záznamy porušující tyto závislosti). To může vést k situaci, kdy

$$M_\infty < M_m \tag{14}$$

Výhodou odhadování struktury dat je nezávislost výsledného modelu na pořadí vstupních dat.

1.4. Příklad

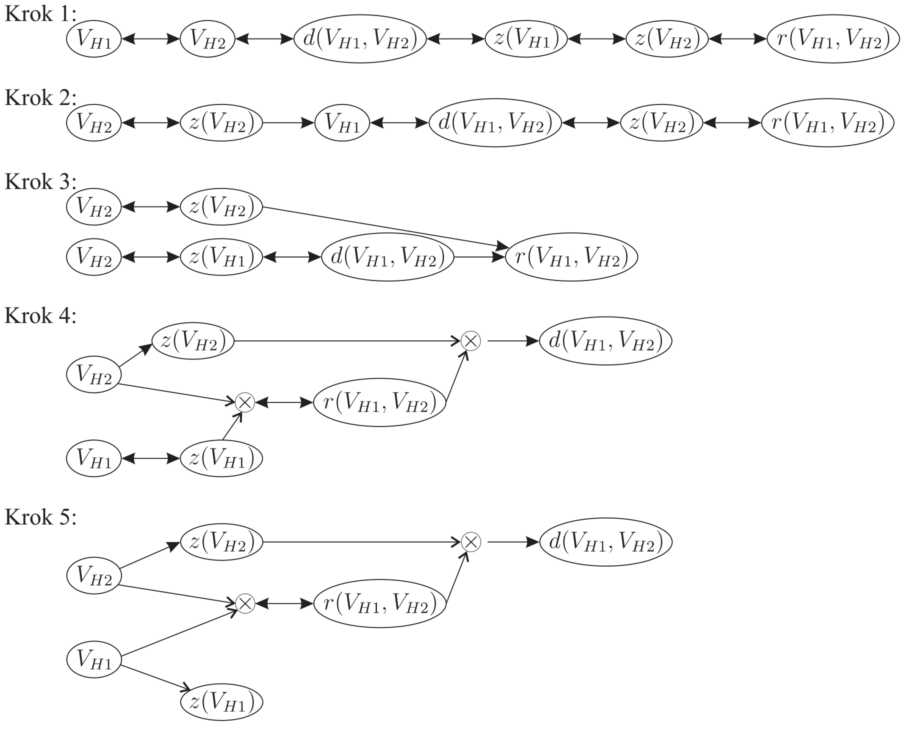
Některé jednoduché úlohy induktivního logického programování lze převést na úlohu odhadování struktury dat [15]. Úlohou induktivního logického programování [16, 17] je najít interpretaci predikátu na základě dat ve znalostní bázi. Příkladem takových algoritmů je GOLEM [18] (využívající postupnou generalizaci), FOIL [19, 20] (postupná specializace), inverzní metody (PROGOL, ALEPH) a další.

Ilustrativní příklad je právě z tohoto prostředí, hledá koncept predikátu $d(V_{H1}, V_{H2})$ popisující skutečnost, že objekt V_{H1} je dcerou objektu V_{H2}. Znalostní báze v tabulce (15) obsahuje predikát $r(V_{H1}, V_{H2})$ (objekt V_{H2} je rodičem objektu V_{H1}) a predikát $z(V_{H1})$ (objekt V_{H1} je ženského pohlaví).

n	V_{H1}	V_{H2}	$d(V_{H1}, V_{H2})$	$z(V_{H1})$	$z(V_{H2})$	$r(V_{H1}, V_{H2})$	$r(V_{H2}, V_{H1})$	$r(V_{H1}, V_{H1})$	$r(V_{H2}, V_{H2})$
1	Eva	Tomáš	$\oplus$	$\oplus$	$\ominus$	$\oplus$	$\ominus$	$\ominus$	$\ominus$
2	Eva	Kamila	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\ominus$	$\ominus$	$\ominus$
3	Milan	Tomáš	$\ominus$	$\ominus$	$\ominus$	$\oplus$	$\ominus$	$\ominus$	$\ominus$
4	Eva	Milan	$\ominus$	$\oplus$	$\ominus$	$\ominus$	$\ominus$	$\ominus$	$\ominus$
5	Karel	Milan	$\ominus$	$\ominus$	$\ominus$	$\ominus$	$\ominus$	$\ominus$	$\ominus$

(15)

Následující obrázek 8 ukazuje vývoj modelu po každém přidání řádku z tabulky (15) podle algo-



Obrázek 8: Ilustrativní příklad

ritmu 1. Je patrné, že koncept predikátu $d(V_{H1}, V_{H2})$ je odhadnut již ve 4. kroce. Model ve smyslu teorie relačních databází můžeme interpretovat tak, že pravdivostní ohodnocení predikátu $d(V_{H1}, V_{H2})$ je funkčně závislé na ohodnocení predikátu $z(V_{H2})$ a $r(V_{H1}, V_{H2})$. Z hlediska modelování struktury dat nezáleží na konkrétních hodnotách ohodnocení, hledáme globální obecný popis vlastností mezi býti-dcerou, býti-rodičem a býti-ženou.

Krok 5 pak ještě doladí koncept predikátu $z(V_{H1})$. Poznamenejme, že model v této fázi žádným způsobem nereflektuje fakt, že různé atributy $z(V_{H1})$ a $z(V_{H2})$ reprezentují tentýž predikát jen s jinou kombinací argumentů.

Použitá vstupní data byla volena tak, aby byla velmi reprezentativní, díky čemuž celý model byl přesně detekován v malém počtu kroků. To ale v praxi u obecných zdrojů nemusí platit.

2. Implementace pomocí pravidlového systému

Metoda byla v první fázi implementována jako soubor uložených procedur v databázovém systému Postgres [5] (verze 7.4). Základní funkčnost byla zajištěna pomocí triggerů. S rostoucím stupněm implementace se však tento způsob stával neschůdným, neboť u metody nejenže závisí na pořadí volání dílčích operací, ale některé z nich pracují již s aktualizovanými údaji a jiné, po nich následující, s údaji před začátkem aktualizace. Z těchto důvodů se stávalo použití databázového systému těžkopádné a SQL dotazy příliš složité. Většina z nich navíc implementovala vztahy ”je-li splněna podmínka, změň data”, tedy prakticky pravidla.

Druhým problémem byla nutnost uchovávat i dočasná data v tabulkách. To lze sice vyřešit použitím dočasných tabulek, avšak toto řešení výrazně zpomaluje start aplikace.

Díky těmto dvěma aspektům se začal rýsovat požadavek na hledání alternativních, lépe vyhovujících, způsobů implementace. Následně byl zvolen přechod na implementaci metody pomocí pravidlového systému.

Jako vhodný kandidát byl posléze zvolen Clips [6] (verze 6.23). Hlavním požadavkem byla možnost dynamicky měnit znalostní bázi (funkční závislosti se postupně odebírají, vznikají jejich nové instance, vzájemně se transformují). Další výhodou tohoto systému oproti jiným je přímá qnávaznost na souborový systém (funkce pro práci se soubory) a dynamické načítání znalostní báze, což v budoucnu umožní paralelní zpracování a uvažování pouze pro výpočet nutných funkčních závislostí a jejich instancí.

Další výhodou je multiplatformita a dostupnost zdrojových kódů tohoto systému, což umožňuje některé složitější nebo v základní sadě neobsažené operace doprogramovat a z ní plynoucí rozšiřitelnost a modularita (např. standardně dostupný modul pro fuzzy pravidla).

2.1. Transitivita, aktualizace kostry

Pro ilustraci použití takového systému pro implementaci metody uveďme několik příkladů vybraných pravidel. Mezi základní operace metody patří aktualizace kostry.

Abychom mohli kostru aktualizovat, je nejprve nutné podle (2) vytvořit koncept transitivity funkčních závislostí.

Pravidlo 1 *Transitivita*

```
(defrule transitivity
          (fd skeleton $?leftside "=>" $?centerside )
          (fd ? $?centerside "=>" $?rightside )
=>
       (assert
          (fd derived $?leftside "=>" $?rightside )))
```

Nadefinujme ve znalostní bázi fakt $(attribute\ order\ \{\ A_1\ \}\ \ldots\ \{\ A_n\ \})$ udávající pořadí atributů a fakt $(attribute\ domaincount\ \{\ A_i\ \}\ |\mathscr{D}_\alpha(A_i)|\ )$ udávající velikost aktivní domény daného atributu. Aktualizace pořadí atributů po přidání nového záznamu do úložiště podle (6) obsahuje pravidlo:

Pravidlo 2 *Aktualizace pořadí atributů*

```
(defrule attribute-order-update
          ?order<-(attribute order $?left { $?attr1 } { $?attr2 } $?right )
          (attribute domaincount { $?attr1 } ?domaincount1)
          (attribute domaincount { $?attr2 } ?domaincount2)
=>
       (if (> ?domaincount1 ?domaincount2)
        (
         (assert
          (attribute order $?left { $?attr2 } { $?attr1 } $?right))
         (retract ?order )
        )))
```

Nyní již máme celý aparát pro aktualizaci kostry. Pravidlo pro případ aktualizace kostry (7) je:

Pravidlo 3 *Aktualizace kostry*

```
(defrule skeleton-update
          (attribute order $? { $?leftside } $? { $?centerside } $? { $?rightside } $? )
          (fd skeleton { $?leftside } "=>" { $?centerside } )
          ?s<-(fd skeleton { $?leftside } "=>" { $?rightside } )
          ?d<-(fd derived { $?centerside } "=>" { $?rightside } )
=>
      (retract ?s ?d)
      (assert
          (fd derived { $?leftside } "=>" { $?rightside } )
          (fd skeleton { $?centerside } "=>" { $?rightside } )))
```

Výhodou tohoto přístupu je jednoduchost pravidel a jejich zápisu, uložené pl/pgsql procedury se stejnou funkčností byly podstatně složitější (kód byl dlouhý řádově v jednotkách kB), k výpočtu optimality kostry (analogicky k pravidlu 3) bylo potřeba spojit v základní verzi celkem 6 tabulek, 3 tabulky obsahující množinu funkčních závislostí a 3 tabulky atributů udávající jejich pořadí (pokud uvažujeme verzi pro komplexní atributy a neredundantní uložení dat, toto spojení je rozšířeno o další 3 tabulky popisující, které atributy tvoří daný komplexní atribut). Navíc na tento dotaz bylo možné provést pouze jednu změnu (dílčí změna kostry ovlivňuje i další závislosti v kostře).

U pravidlového systému se jedná o nalezení 4 faktů ve znalostní bázi, problematický může být jen počet možných kombinací, který je ale u těchto systémů dobře zvládnutý. Druhou možnou kombinací je interpretace negace výskytu faktu, které opět může vést k velkému počtu přípusntných kombinací.

2.2. Úložiště

Naopak některé operace se při použití pravidlového systému nepatrně komplikují. Jednou z nich je operace přidávání relace do úložiště, kterou nelze provést přímo, ale oklikou přes vzor takové instance. Mějme pro každý atribut nadefinován fakt ve tvaru $(attribute\ pattern\,\{\ A\ \}\ \{\ [\ A\ \_\ ]\ \}\ )$ nebo v případě komplexního atributu $(attribute\ pattern\,\{\ A\ B\ \}\ \{\ [\ A\ \_\ ]\ [\ B\ \_\ ]\ \}\ )$ a vkládaný záznam rozdělený po atributech $(tuple\ attribut\ hodnota\ )$.

Vložení záznamu do úložiště představuje nejprve sestavení vzoru instance funkční závislosti:

Pravidlo 4 *Příprava vzoru instance*

```
(defrule prepare-instance
          ?p<-(fd skeleton { $?leftside } "=>" { $?rightside } )
          (attribute pattern { $?leftside } { $?leftpattern } )
          (attribute pattern { $?rightside } { $?rightpattern } )
=>
      (assert
          (rel ?p { $?leftpattern } "=>" { $?rightpattern } )))
```

V druhém kroku je pak tento vzor naplněn daty podle vkládaného záznamu.

Pravidlo 5 *Plnění daty*

```
(defrule data-fill
          ?r<-(rel ?p $?left [ ?attr _ ] $?right )
          (tuple ?attr ?value] )
=>
      (retract ?r )
      (assert
          (rel ?p $?left [ ?attr ?value ] $?right )))
```

Výsledkem je uložený předpis pro asociační pravidlo.

3. Shrnutí

Příspěvek volně navazuje na práci publikovanou v minulém ročníku Doktorandského dne [21]. Základní myšlenka řešení problematiky zůstává stejná, dílčím způsobem se modifikují cíle práce.

Oproti předchozímu ročníku není hlavní cíl spatřován ve fuzzifikaci funkčních závislostí, neboť porušení klasické funkční závislosti v reálných datech může být problematické (data nemusí být natolik reprezentativní, aby došlo k porušení závislosti), tudíž další oslabení této vlastnosti není žádoucí. Spíše tedy dochází ke kladení požadavků na vstupní data, předpokládají se deterministické hodnoty, bezchybnost a konzistence zdrojů dat a hledání alternativních procesů, např. problémy chybovosti se dají převést na známý problém hledání vyjímek asociativních pravidel.

Hlavním cílem naopak zůstává orientace na sémantický web, konkrétně odhad sémantiky ze struktury a popis integrace modelů. Díky možnosti použít pravidlových systémů pro implementaci metody a s tím související implementací jednoduchých pravidel převádějící data na jiné reprezentace či různé granularity primitivních vztahů (např. pravidla pro získávání metadat nebo jednoduchá kombinace různých typů vztah; při prohledávání úložiště). Tyto nové možnosti otvírají cestu k jednoduššímu popisu vztahů mezi atributy, zjednoduší se popis integračního procesu (bude postačující přidat nový typ vztahu mezi atributy a příslušné odvozovací pravidlo).

Novým cílem se stává hledání generalizovaného modelu umožňující odhadování hodnot atributů na základě vlastností zintegrované kostry několika modelů, principy integrace či hledání výjimek. Poměrně zajímavým tématem se jeví konstrukce metadat, konverze mezi různými interpretacemi informace (různá granularita atributů majících tentýž význam) nebo hledání primitivních částí informace a vztahů mezi nimi.

Za poměrně úspěšné lze považovat hledání metodických isomorfismů. Podařilo se ukázat, že některé základní úlohy induktivního logického programování, jejichž zadání splňuje požadavky metody odhadování struktury dat, lze na tuto metodu převést [15]. Tento převod buď lze udělat triviálním způsobem transformováním znalostní báze do tabulky pokrývající příslušné kombinace konstant, jak ukazuje příklad (15), nebo (existuje-li taková informace - ohodnocení predikátu je závislé na jeho proměnných nebo na ohodnocení jiných predikátů) znalostní bázi převést přímo do relačního modelu a počítat pouze závislosti kolem atributu reprezentujícího hledaný predikát.

Další oblastí je rozbor vlastností kostry a její různé konfigurace (viz obrázek 1, 2) a definice uspořádání atributů (6) a vliv tohoto uspořádání na kostru (viz obrázek 3, 4). Zde se jeví jako perspektivní přístup postupného (proudového) on-line zpracování dat. On-line zpracování je vhodné jak z hlediska pravidlových systémů, tak z hlediska možné paralerizace problematiky. Postupné zpracování vychází z možnosti polynomiálně složité iterační aktualizace kostry oproti NP úplnému hledání jádra z transitivního uzávěru.

V neposlední řadě došlo oproti [21] k rozšíření modelu o komplexní atributy a analýze problému vytváření komplexních atributů v modelu (obrázky 5, 6, 7) a hledání netriviálních funkčních závislostí kolem těchto atributů [4]. To je jedinou částí vykazující až na speciální případy (10) nepolynomiální složitost (9).

Metoda byla implementována původně jako uložené pl/pgsql procedury databázového systému pro variantu, kdy data jsou uložena redundantním způsobem v předem dané struktuře (prakticky reprezentující jednu relaci mezi zadanými atributy). Tato varianta pokrývá celou problematiku odhadu struktury dat. Varianta s daty ukládanými podle odhadnutého modelu je rozpracována, některé partie se ukázaly jako obtížně zvládnutelné korektním způsobem. Proto došlo k přerušení implementačních prací a migraci z databázového systému na systém pravidlový, který se pro implementaci metody v současné době jeví jako velmi perspektivní. V současné době probíhá implementace metody právě v tomto systému (viz pravidla 1 až 5).

Jelikož znalostní báze generovaná metodou na základě odhadnutého modelu je ve formě asociačních pravidel, bude zajímavé do takové báze přidat znalosti pocházející z dokumentů sémantického webu a sledovat vliv struktury těchto dokumentů na globální model. V budoucnu by měla být data z této znalostní báze včetně metadat přístupná přes webové rozhraní a mělo by být implementováno webové rozhraní pro vyhledávání dat ve znalostní bázi.

Literatura

[1] Grigoris Antoniou, Frank van Harmelen. “A Semantic Web Primer”. MIT Press, 2004. ISBN: 0-262-01210-3.

[2] Eric Miller, Ralph Swick, Dan Brickley. “Resource Description Framework”. <http://www.w3.org/RDF/> [on-line]. 2004.

[3] Eric Miller, Jim Hendler. “Web Ontology Language”. <http://www.w3.org/2004/OWL/> [on-line]. 2005.

[4] Martin Řimnáč “Web Information Integration Tool - Data Structure Modelling”. In *Proceedings of 2005 International Conference on Data Mining*. CSRea, USA. pp 37-40. ISBN 1-934215-79-3. 2005.

[5] Postgres. <http://www.postgres.org/> [on-line]. 2005.

[6] Clips. <http://www.ghg.net/clips/CLIPS.html> [on-line]. 2005.

[7] G. Grahme, K. Räihä, “Database Decomposition into Fourth Normal Form”. In *Conference on Very Large Databases*. pp. 186–196, 1983.

[8] G. Ausiello, A. D’Atri, M. Moscarini, “Chordality Properties on Graphs and Minimal Conceptual Connections in Semantic Data Models”. In *Symposium on Principles of Database Systems*. pp. 164–170. 1985.

[9] Bruno T. Messmer, Horst Bunke ”Efficient Subgraph Isomorphism Detection: A Decomposition Approach”. In *IEEE Transactions on Knowledge and Data Engineering.* pp.: 307-323. 2000.

[10] F. Cuppens, K. Yazdanian, ”A Natural Decomposition of Multi-level Relations”. In *IEEE Symposium on Security and Privacy*, pp. 273-284. 1992.

[11] B.N. Shamkant, R. Minyoung, “Vertical Partitioning for Database Design – a Graphical Algorithm”. In *SigMod*, pp. 440–450, 1989.

[12] L. Bellatreche, K. Karlapalem, A. Simonet, “Algorithms and Support for Horizontal Class Partitioning in Obect–Oriented Databases”. In *Distributed and Parallel Databases, 8*, Kluwer Academic Publisher. pp. 115–179, 2000.

[13] A.A.Barfourosh, M.L. Anderson, H.R.M.Nezbad, D Perlis. ”Information Retrieval on the World Wide Web and Active Logic: A Survey and Problem Definition”. In <http://citeseer.ist.psu.edu/barfourosh02information.html> [online]. 2002.

[14] C. Beeri, P.A.Bernstein, ”Computational Problems Related to the Design of Normal Form Relation Schemes”. In *ACM Transactions on Database Systems*. 4,1. pp 30-59. 1979.

[15] Martin Řimnáč, ”Odhadování struktury dat a induktivní logické programování”. In *ITAT 2005* . (V tisku). 2005.

[16] Nada Lavrač, Sašo Džeroski, “Inductive Logic Programming - Techniques and Applications”. Ellis Hordwood, Chichester. ISBN: 0-13-457870-8. 1994.

[17] Sašo Džeroski, Nada Lavrač, “Relational Data Mining”. Springer-Verlag, Berlin. ISBN: 3-640-42289-7. 2001.

[18] Muggeton, S., Feng, C., “Efficient introduction of logic programs”. In *Proceeding of the First Conference on Algorithmic Learning Theory*. pp 368-381. 1990.

[19] Quilan, J., “Learning logical definitions from relations”. In *Machine Learning*, 5(3). pp 239-266. 1990.

[20] Quilan, J., “Knowledge acquisition from structured data - using determinate literal to assist search”. In *IEEE Expert*, 6(6). pp 32-37. 1991.

[21] Martin Řimnáč, ”Rekonstrukce databázového modelu na základě dat (studie proveditelnosti)”. In *Doktorandský den ’04, Ústav informatiky AV ČR*. pp. 113-120. ISBN 80-86732-30-4. 2004.

Sharing Information in a Large Network of Users

Post-Graduate Student:
ING. ROMAN ŠPÁNEK

Department of Software Engineering
Faculty of Mechatronics, TU Liberec
Hálkova 6
461 17 Liberec

Czech Republic

roman.spanek@vslib.cz

Supervisor:
ING. JÚLIUS ŠTULLER, CSC.

Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2

182 07 Prague 8

stuller@cs.cas.cz

Field of Study:
electrical engineering and informatics
Classification: 2612v045

Abstract

The paper[1] describes a possible treatment of data sharing in a large network of users. The mathematical model is based on weighted hypergraphs whose nodes and edges denote the users and their relations, respectively. Its flexibility guarantees to have basic relations between users robust under frequent changes in the network connections. Approach copes with the communication/computing issues from different point of view based on a structure evolution and its further optimization in sense of keeping the *parallel* space and time complexities low. Although the idea is aimed to the field of mobile computing, it can be generalized in straightforward way to other similar environment. An experimental application is also proposed and discussed in the paper.

1. Introduction

The aim of the paper is to present a consistent model of a reconfigurable network with a distributed management useful for representing security and other relationships among users and their groups. By the *consistency* of the model we mean that the internal structure of the network must not degenerate over time to its limiting cases: small number of very large groups, or large number of small groups. In other words, our model has to have a feedback related to the fragmentation of the network into subnetworks. This we will achieve by careful use of our mathematical model, related algorithms, and implementational details.

The mobile computing area (see Figure 1) is faced by some natural limits like small bandwidth, battery power, and also by the needs of communication and computation flexibility [2],[3],[4]. While we can still expect rapid improvements in the areas of these limitations, an important question is the security of wireless networks [5]. In recent years there has been a strong progress in development of coding and cryptography which allow to have pretty secure individual transmissions. On the other hand, the security on a higher level of abstraction is still an open question and has to be addressed [6],[7],[8]. Namely, it is important to have tools for defining, evaluating, and maintaining concepts of group securities. This assumes creating an infrastructure of a network where users are restructured into smaller units (groups), and they take into account these relations in their communication [9],[10].

[1]The work was partially supported by the project 1ET100300419 of the Program Information Society (of the Thematic Program II of the National Research Program of the Czech Republic) "Intelligent Models, Algorithms, Methods and Tools for the Semantic Web Realisation".

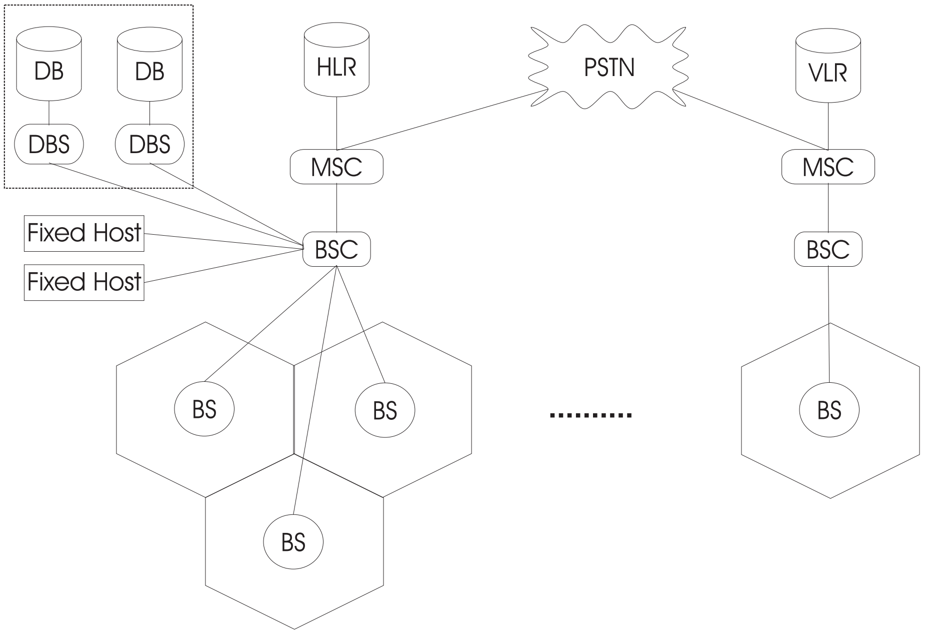


Figure 1: Mobile Database Architecture

This paper proposes a full model which can handle security issues in mobile networks which can be dynamically evolving, or frequently reconfigured. Its mathematical basis form weighted hypergraphs. Edge and vertex weights can express most practical situations related to users (vertices) and relationships (hyperedges). By the term practical situations we mean not only group security issues but also possible generalizations. Network reconfigurations induce new hypergraph weights. We propose algorithms for local modification of weights. This is important for distributed runs of the algorithms. The choice of the algorithms guarantees that the model will not degenerate to a steady-state limiting case with too many or too little groups. Dynamic changes in the hypergraph model can be implemented using standard tools from numerical linear algebra [15].

The paper is organized as follows. Section 2 describes the mathematical model. Basic operations on underlying structure are proposed in Section 3. Section 4 is devoted to application of the model and the following section 5 is sharply aimed to make a brief overview on the real implementation of proposed algorithms. The paper finishes by some conclusions.

2. Mathematical Model

Our mathematical model is based on weighted hypergraphs with general real weights. Note that in many cases we will use integer weights only. In the following we will introduce some basic terminology.
The weighted hypergraph is a generalization of the concept of weighted graph which allows edges incident to more than two vertices. Formally, hypergraph $\mathcal{H}$ is a quadruple (V, E, W_V, W_E), where V is its set of vertices, $E \in 2^V$ is its edge set, and W_V and W_E present vertex and edge weights, respectively. That is, the weights are mappings from V and E to the set of reals, respectively. For simplicity, we will consider $V = \{1, \ldots, n\}$. Note that in some applications the incidence of vertices and edges is explicitly expressed by so-called vertex and edge connectors [11]. Here we prefer the classical definition [12] given above which is suitable for our purposes. Most of the important concepts from graphs can be easily generalized to hypergraphs. Here we will mention only some of them. The weights may express further properties of graphs and hypergraphs, in addition to their connections. For instance, they can express quality or priority of communication, cost issues, efficiency of evaluations in individual vertices. As we will emphasize later, in our application related to security in networks, the vertex weights will express *reliability* of users, and edge weights correspond to *security of vertex connections*.

3. Basic Operations on Hypergraphs and Related Data Structures

This section is devoted to a brief overview of basic operations used in our application, and underlying data structures. Note that application-related issues will be treated later.

A basic assumption of the mobile computing and/or communication is that the whole hypergraph model is not centralized but distributed. In addition to global network properties we need to store most of the data locally, or in close neighborhood of vertices. Small to medium grids may assume that individual vertices store: a/ vertex related information, b/ their adjacency sets c/ weights of incident (hyper)edges depending on the size of their local memories. If the local memory allows it, they may store also more levels of information. They can store vertices and edges up to their k-adjacencies for some $k >= 1$ which is what we will assume. Larger grids may need a domain-based distribution [13] for faster execution/communication even if the local memories are relatively large. Specific vertices (group leaders) contain all relevant information on weights and incidences for whole domains. In our implementation they correspond to meta-vertices, i.e. seed vertices of meta-associations related to groups of users. The following text describes some application related issues which help to reflect specific model features.

4. Application

A crucial assumption is that the application is distributed. That is, the representation and the updates must be implemented with respect to this. Therefore, all the proposed ideas have to keep the *parallel* space and time complexity low.

4.1. Group representation

Two extremal types of distributed representation of groups (subgraphs) are as follows. First, a user (note that a user is equivalent to a vertex in the application section) stores all information (connections, edge weights) related only to its adjacency set (set of all its neighbors). In this case, communication with neighbors has time complexity O(1), space complexity is small, but overall communication with distant neighbors might be expensive. Second, users can store more levels of neighbors (for the concept of levels in graphs which can be easily generalized to hypergraphs see [14]). That is information on neighbors of neighbors can be stored, and so on. Using more explicit representation of more distant vertices decreases average time complexity to communicate with other users but (local) space complexity may be rather high. Complexities are described more in detail in section 4.5. Our representation is a hierarchical with more *layers* of hierarchy. The *base layer* is formed by standard users. Each group has a representant that is responsible for group management called a *Group leader* (GL). This vertex is also responsible for communication support to other groups. The group leaders form the *enhanced layer* of users. Communication is allowed only between nodes on the same level or to one level deeper. This combines advantages of both extremal possibilities. The vertices are members of groups which dynamically evolve depending on changes in vertex and edge hypergraph weights. A group joining process is managed by a special node, called *Group Gate* (GG).

An important assumption for distribution of vertices into groups is that strong hypergraph components will be a part of the same group. That is, the connections among vertices form an *acyclic* hypergraph.

4.2. Edge evaluation

Up to now we have explained the role of vertex weights. Edge weights are the basis of the dynamic behavior of the network of users. In other words, by their appropriate evaluations and reevaluations the network thus gets its *Social Network Property*, i.e. the ability to describe social relationships in terms of a weighted hypergraph model. Consequently, in this subsection we will present a consistent set of rules and algorithms which do represent consistent static and dynamic properties of a secure network. The fact that an edge has a high weight we will equivalently call that the connected vertices (users) have a good mutual relationship.

4.2.1 Edges' evaluation rules: Here we will describe possible cases which force evaluation and reevaluation of hypergraph edges. The procedures can be applied in all layers of vertices of a hierarchical model but we will not distinguish this here. Although we have taken into consideration full spectrum of

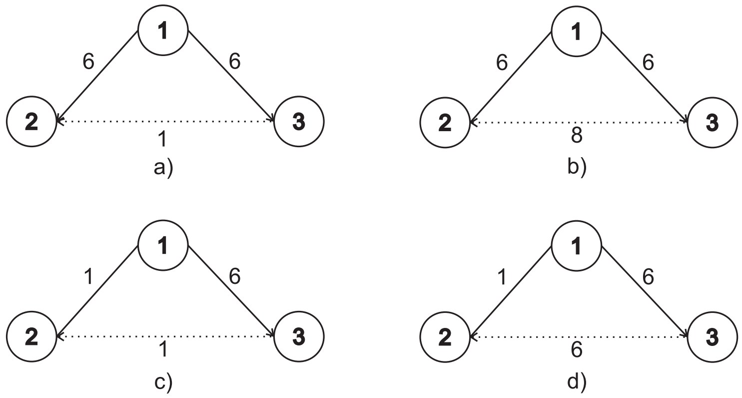


Figure 2: Four possible cases.

possible modifications in edge reevaluation and addition, one which worths mentioning here is a case depicted in Figure 2. A newly added edge is plotted in a dotted line. When the edge has been added into the structure its weight might by in imbalance with the old ones (Figure 2 d)). If so there is a need to do some reevaluation that will preserve the weights in the consistent state. A possible approach is to decrease the weight between vertices 1 and 3 to the weight of edge 1-2. Another possibility is to give to all vertices in the cycle their average value. The decision can be made automatically based *on user trigger* (see section 4.4) or chosen by involved users.

Once edge has been re-evaluated, it might cause some other re-evaluations. Consequently, this might result in huge computational overhead. In cases a), b), and c) we do insist that the further re-evaluation is not necessary. On the other hand, in d) this is not longer true. Because of imbalance in edges' weights (namely edges 1-2, 1-3), re-evaluation has to be done to preserve security properties of the structure based on mutual relationships. When the new edge has weight 6, the weights of edges (1-2, 1-3) must be re-evaluated. Once the weights were changed we are done since we get case a) or c).

4.3. Optimization and self-monitoring

We will introduce two important terms *graph condensation* and *graph expansion*. *The graph condensation* means representing groups of nodes by a single item. It frequently occurs when new nodes and/or new edges are added as shown in the previous Section 4.2.1.

Graph expansion will be performed if a group of users fulfills $|K| > \delta\,|K_{avg}|$, where
$|K|$ is the component size, δ is a non-negative real number, and $|K_{avg}|$ is the average size of all groups of users in the structure. In this case, the component is too big and cannot be efficiently managed, and a graph expansion is initiated. It divides the group into a set of smaller ones. For each newly created component a GL is found and necessary data structures are set up as mentioned in section 4.2.1.

For optimization and diagnosing purposes, additional parameters are defined. The one which is worth mentioning here is a *time stamp* that stores the last moment when the edge was used. It helps to evaluate an importance of the edge. At specified time moments, the group leader runs a program structure *Trigger* (see section 4.4) to get a difference between the time stamp and the real time, given as $Time\_Diff = Time\_Stamp - Actual\_time$. If the size of $Time\_Diff$ is too small (with respect to a threshold), the edge is deleted.

4.4. Triggers

Our implementation makes use of triggers in order to evaluate and reevaluate the role of users in the network. In fact, this is a global communicating mechanism which keeps the overall consistency of our distributed model. The most important role of triggers is to check whether the users' weights (vertex weights) and weights of their connections (edges and hyperedges) are consistent. In our case, whether they express a consistent model with weighted securities of users and their connections. On *a scheduled trigger* the structure is inspected. Group leader issues an *optimization message*, which propagates through the structure and users optimize their adjecent sets in the terms of space utilization and computation load. An *on event trigger* is run whenever a new edge has been added into the structure. The action is reported to the corresponding GL by *an information message*.

An *on user request trigger* is a way how a user can influence the hypergraph structure more globally. A user can predefine some actions which should be treated differently then defined for the whole group. Another reason to start this trigger may be the need to modify asset numbers of some of users which influence the user. In the other words it enables a personalization in the structure.

4.5. Sharing

This subsection explains some issues concerning sharing and communication among users. Communication is done through message interchanging. The *target user id* is the unique group identifier of the user that is asked for data. Second value identifies the data demander. Demander asset number and edge's weight are also included. The *link type* cell holds information either it is transitive or direct, over-board or inter-group link respectively.

As given above, each user may store up to k level of adjacent neighbors. Clearly, if $k = 1$ then the communication complexity

$$2 * d(K), o \tag{1}$$

where $d(K)$ is a diameter of the group K (which may consist of more components of the acyclic hypergraph). Space complexity is

$$|E| = \sum_i d_{out}{}^{(i)}. \tag{2}$$

If $k = 2$, communication load is

$$d(K) \tag{3}$$

and space complexity is

$$|E| = \sum_{(i,j) \in E(K)} d_{out}{}^{(i)}.d_{out}{}^{(j)}. \tag{4}$$

Each user has a *sharing value* set for sharing data. In our case, we use an integer value denoting a threshold under which sharing information with other users is rejected.

The following three rules controlling the sharing feature are defined in our implementation. The first rule is called *Direct*. It is useful in cases where data sharing is requested by users from an adjacent set. In such case direct connection weight and user's asset number are taken and compared to sharing value. The second rule will be called *friend-of-my-friend*. In such case there were no direct connections between users marked 1 and n. Therefore the message was released and users were forwarding it since the destination, user n, was found. Consequently the minimal edge's weight of all possible paths is taken and compared to sharing threshold. The third rule we call the *over-border* rule. This rule is based on the communication via appropriate group leaders. This procedure offers a possibility to join groups with common interests from different groups.

4.6. Security questions

Up to now we have discussed mostly mathematical and technical questions of a network of users. There are some global security questions which should be satisfied by any useful model. One of the biggest threat in security area is misapplication of private information (e.g. credit card number, personal identification number). Namely, if some secure information have been stolen from their proper holder, it can be consequently misused. Potential damages can be minimized by a time stamp validation. Since time stamps and invalidation times are part of GIN, both the proper user and the group leader/gate know time of the GIN invalidation. Further, the proper user and group leader/gate expect the time of the GIN invalidation and they also anticipate issue of a new GIN. Therefore only very rarely a user with stolen GIN, would try to access group with invalid GIN and can be easy revealed. The main question, which still remains, is how to treat with the GIN misuse while it is still valid? A possible solution lies in the underlying hypergraph structure. The structure is under continual evolution and it is optimized on every question issued by a group member (see section 4.3). Therefore the structure reflects favorite relationships between users. An example would be a good way to make it clear. Assume that a user stole a GIN and he/she is about to download as most as she/he could till the stolen GIN become invalid. In such case the "bad" user would ask as many group participant as he/she could and would demand data. The hypergraph structure - however, reveals such misuser very quickly. Once a group member behavior is found very different from its standard behavior, the user is disconnected from the network and a new GIN is consequently issued to the proper user. The time stamp validation time and the behavior check based on the underlying graph structure will significantly improve security issues.

5. Experimental Implementation

As the precedent was manly devoted to theoretical issues related to communication/computing in the large network of users, the implementation section makes the proposals alive. While the target environment might change a lot, as was mentioned earlier, an experimental application, called SECMOBILE, must be designed and created with respect to this. ANSI C language offers us a great opportunity to re-use existing implementations of both numerical and distributed computing paradigm with very optimized code as an imported DLL. Although the application is currently written as a console application without any graphic interface, in the future it will be given by a user friendly one.

The aim of the experimental application is to prove the correctness of the proposed algorithms and refine them if necessary. By the prove we will consider behavior of the network to not degenerate into one of the limiting cases; many groups consisting only one node; or few groups covering all nodes.

The implementation is based on a dynamic system of structures describing non zero values in the "adjacency" matrix. Adding new edges and vertices into the existing structure is maintained through binary or text files. Text files are designed in CSV fashion with a space separator. Since the structure should reflex usual behavior of users, creating such network cannot be held by random generation of nodes and/or vertices. An input file should be well formed taking into consideration "probable behavior" of a user.

While the proposed methodology keeps into consideration the complete spectrum of possibilities, the experimental application employs only a part. Its target is a structure creation and related operations. Higher levels of abstraction (e.g. sharing, user invitation process) will be considered and added consequently.

6. Conclusions

The paper deals with the security issues from a viewpoint of a distributed and self-evolving application. The design is mathematically sound being based on the weighted hypergraph model. The low level algorithms use supernode merging and splitting which was implemented in numerical linear algebra. The model is distributed and the local space and time complexities are very low. We discusses the relation among graph items and real-world application. The most important question whether the model can be consistently developed and model secure communication among users and their groups was answered positively in our implementation. Although we explained the model in the environment of mobile computing and communication, it can be easily generalized for some other application areas.

References

[1] S. Nanda, D.J. Goodman, "Dynamic Resource Acquisition in Distributed Carrier Allocation for TDMA Cellular Systems", *Proceedings GLOBECOM*, pp. 883–888, 1991.

[2] S. DasBit and S. Mitra, "Challenges of computing in mobile cellular environment a survey", *Elsevier B.V.*, 2003.

[3] A. Flaxman, A. Frieze, E. Upfal, "Efficient communication in an ad-hoc network", *Elsevier*, 2004.

[4] S. Basagni, "Remarks on Ad Hoc Networking", *Springer-Verlag*, Berlin Heidelberg, 2002.

[5] R. Molva, P. Michiardi, "Security in Ad Hoc Network", *IFIP International Federation for Information Processing*, 2003.

[6] Y. Lu, B. Bhargava, W. Wang, Y. Zhong and X, Wu, "Secure Wireless Network with Movable Base Stations", *IEICE Trans. Community*, vol. E86-B, 2003.

[7] Y. Zong, B. Bhargava and M. Mahoui, "Trustworthiness Based Authorization on WWW", *IEEE Workshop on Security in Distributed Data Warehousing*, 2001.

[8] J. Park, R. Sandhu and S. Ghanta, "RBAC on the Web by Secure Cookies", "Database Security XIII: Status and Prospects", Kluwer 2000.

[9] R. S. Sandhu, E. J. Coyne, H. L. Freinstein and C. E. Youman, "Role Based Access Control Models", *IEEE Computers*, Volume 29, 1996.

[10] P. K. Behera, P. K.Meher, "Prospects of Group-Based Communication in Mobile Ad hoc Networks", *Springer-Verlag Berlin Heidelberg*, 2002.

[11] P. O. de Mendez, P. Rosenstiehl, P. Auillans and B. Vatant, "A mathematical model for Topic Maps", *Springer-Verlag*, Berlin Heidelberg, 2002.

[12] M.C. Golumbic, "Algorithmic graph theory and perfect graphs", *Academic Press*, 1980.

[13] N. Selvakkumaran and G. Karypis, "Multi-objective hypergraph partitioning algorithms for cut and maximum subdomain degree minimization", *IEEE Transactions on Computer-aided design*, 2005, to appear.

[14] A. George and J.W.H. Liu, "A fast implementation of the minimum degree algorithm using quotient graphs", *ACM Trans. Math. Software*, 6(1980), 337–358.

[15] A. George, J.W.H. Liu, "Computer Solution of Large Sparse Positive Definite Systems", *Prentice Hall*, 1981.

Electronic Health Record and Telemedicine

Post-Graduate Student:
MGR. JOSEF ŠPIDLEN
EuroMISE Centrum – Cardio
Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2
182 07 Prague 8
spidlen@euromise.cz

Supervisor:
RNDR. ANTONÍN ŘÍHA, CSC.
EuroMISE Centrum – Cardio
Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2
182 07 Prague 8
riha@euromise.cz

Field of Study:
Biomedical Informatics
Classification: 3918V

This work was supported by the project no. 1ET200300413 of the Academy of Sciences of the Czech Republic, and by the Institutional research plan no. AV0Z10300504.

Abstract

According to the objectives of the Ph.D. thesis I have studied the possibilities of electronic health record representation and analyzed the suitability of various data storing techniques. On the basis of the EHR system requirements, e.g. the dynamically modifiable set of collected concepts, I have picked up the threads of my diploma thesis, I have further extended and mathematically formalized them during the work on this Ph.D. thesis and during the research and development in the frame of my employment at the Institute of Computer Science, Academy of Sciences of the Czech Republic, the EuroMISE Centre. I have proposed a completely new *Graph Technology of Medical Knowledge and Healthcare Data Structuralization* that was submitted as an application of invention with a patent request to the Czech Industrial Property Office. I have implemented the stated technology as a data basis of the MUDR electronic health record (EHR) that this paper describes in detail.

Although the MUDR EHR meets practically all the requirements stated for an EHR system, the real practice showed that there are situations when there should be an effortless solution that could be more customized, e.g. for physicians collecting special kind of data for the purpose of a clinical study. As a response to this need I started with research and development of a lighter form of an electronic health record called MUDRLite that provides a functionality to extend the potential and capabilities of the system including advanced user-defined components. The corresponding interface was tested by integrating the *Dental Cross* custom component, whose data model is based on the *Dental-Medicine Data Structuralization Technology Using a Dental Cross* that was also applied for a Czech national patent.

Last but not least, I considered the possibilities of using classical mobile phones as a telemedicine applications' platform. I have analyzed the possibilities of the Micro-Edition of the Java language with the focus on its usage for telemedicine purposes as also described in the paper.

The thesis was submitted with the date of defence set to October 4^{th}, 2005.

1. Introduction

Nowadays, in the medicine area there is still a number of problems and unsolved issues. Many computer scientists keep on trying to find new possibilities of solving various tasks to make the computers serving in the healthcare more effective. The advancement of medical informatics was raised by the ongoing specialization of medical professions and thus the need of sharing information about patients. The development in medicine motivates significantly the usage of information technologies for the purpose of mass data processing in the medical domain. New research topics are emerging, e.g. the optimal representation of a patient record, of medical documentation or of medical knowledge and guidelines.

Medical informatics is understood as a specialization providing complex information services in the public health domain. The importance of this field is increasing constantly as the computer scientists are getting into a new position as a part of a top management of hospitals, pharmacies and health insurance companies. Together with the importance of medical informatics its field of activity grows proportionally. Specific requirements are set on applications in the field of *hospital information systems*, *healthcare registers*, *telemedicine*, etc.

Recently, the *electronic health record* (EHR) has become a crucial part of medical documentation. Its implementation, with a stress on maximally structured information in conjunction with other tools, becomes a basis for *personalized health care* based on *evidence*, *knowledge* and *medical guidelines*. The new possibilities of clinical data sharing have the potential to cut down the number of pointlessly repeated examinations and thus to lower not only the strain on patients but also the expenditures in healthcare. The use of *telemedicine* even enhances those benefits by adding new possibilities of distance care and consultation.

2. The State of the Art

2.1. EHR Standards and Norms

Relevant standards and norms are mainly created by international organizations operating in the health informatics. Any discussion on EHR standards requirements must begin by defining not only the EHR as an entity but also the scope of what constitutes an EHR standard. One viewpoint is that the job of groups developing standards for the EHR is limited to the structure and function of the record and systems processing the record. This is inherent in the structure of many health informatics standards organisations, including *ISO/TC 215*, which typically divide standards working groups into the *EHR*, *messaging*, *terminology and concept representation*, and *security issues*. A broader viewpoint is that EHR standards include standards for all of the EHR building blocks, i.e. the *EHR structure*, *terminology*, *messaging*, *security*, *privacy*, etc.

By far the most important components of healthcare data are clinical data that are directly related to the patient care. Unfortunately, there is currently a massive fragmentation of them. This contributes significantly to the cost of information management, but more importantly, it leads to a lower quality of patient care, ant to more medical errors as well. The focus of EHR standardisation should therefore be to promote a high level of interoperability of clinical information systems within healthcare organisations and between healthcare organisations, initially within individual regions and countries, but global interoperability of EHR systems should be the ultimate aim. Moreover, in order to support future automatic processing and functions like intelligent decision support, the aim must be to build EHR standards, which support not just functional (*syntactic*) interoperability but full knowledge-level (*semantic*) interoperability as well [1].

The major problem is the huge amount of different proprietary or standardized interfaces [2], e.g. *message or interface standards* like HL7, EDIFACT, DICOM, rather *content oriented standards* like LOINC, ICD-10, ICPM or hybrid approaches like CEN 13606, openEHR to name but a few. However, none of the standards is sufficient to cover completely the EHR issues to achieve the high level of EHR interoperability.

2.2. EHR Projects and Best Practices

The research in the field of electronic health record is a matter of great concern to many institutions and working groups. Within the *SynEx* (*Synergy on the Extranet*) project, partners from nine European countries were developing a secure and shareable electronic health record. They used the HTTPS protocol for data exchange purposes and XML for data encoding using the Synapses Object Model, which is similar to the CEN ENV 13606 model. As a semantic for the collectable attributes they used so-called *archetypes* - formal models of clinically relevant EHR elements defining their data structure and terminological basis. The research continued under the patronage of the *openEHR Foundation* and it has defined the so-called *Good Electronic Health Record* (GEHR), which served as an information source for my further work.

Thanks to my employment in the EuroMISE Centre I have learnt the results of the *I4C/TripleC* project where the EuroMISE Centre had cooperated in the past. The I4C/TripleC has developed the *ORCA* (*Open Record for Care*) EHR [3] that also influenced my further work.

2.3. The Main Objectives of the Thesis

According to the assignment the main objectives of this thesis are stated as follows:

- To analyze the suitability of various techniques of electronic health record data representation including multimedia attributes, to evaluate the appropriateness of XML-based trends for communication and other purposes and to consider the possibilities of EHR systems design including the integration of decision support systems in the form of formalized medical guidelines.
- To evaluate the EHR possibilities in the telemedicine area including the possibilities of remote access to EHR systems using mobile devices.

Because of the applied character of this research, pilot implementations of selected software components should become a part of the thesis.

3. Author's Research and Development

3.1. MUDR Electronic Health Record

3.1.1 Architecture of the System: The system is based on a 3-tier architecture with a database layer, an application layer and a user interface layer. The function of the database layer is to store the data and check basic data integrity. The application layer provides a view of the data connected with the corresponding context and without implementation details. The user interface layer represents various client-applications designed mainly for physicians to view and manipulate patients' data. Another type of a client application can be an automated system making statistical processing of the data.

Within the implementation I have split the application layer furthermore. The application layer service communicates directly with the database layer, but it does not communicate directly with user interfaces. There is an XML-based messaging between the user interface layer and the application layer and thus it is advantageous to use the HTTPS protocol to ensure the communication. It would be a pointless effort to develop an own HTTPS server and thus a third-party HTTPS server can be used (e.g. the Apache2 web server).

Optional components of the MUDR EHR system are libraries of formalized *medical guidelines*. These guidelines represent decision support tools integrated to the application layer, however, medical guidelines are updated continuously and thus they should not be integrated to the application layer service tightly. From a special point of view a formalized guideline can be understood as a special type of EHR client application that acquires patient's data. From another point of view it can be understood as a server application that provides a kind of services (its advice) based on the patient's data combined with medical knowledge hardwired in a formalized way. Thus, within the pilot implementation I have designed medical guidelines as dynamic libraries (DLL) that can be linked dynamically to the application layer service. The set of the libraries can be updated continuously without any need of the service recompilation. The same MUDR API is used for the communication between the application layer service and medical guidelines as for the communication with user interfaces. As a pilot test guideline I have formalized and implemented the *1999 WHO/ISH Guidelines for the Management of Hypertension* [4].

3.1.2 Data Representation: The main goal of my work was to suggest common general principles to increase the quality of EHR systems, to simplify data sharing and data migration among various EHR systems and to help to overcome the classical free-text based health record. I did not want to choose a particular database or an operating system and thus I tried to propose an open information storage meta-model with various implementation possibilities as inspirations for EHR software vendors.

Because of the requirement of a dynamically extensible and modifiable set of collected attributes, it is complicated to use a classical relational database structure with columns corresponding to the gathered features as the basis of the information storage. The data representation in the MUDR EHR application uses an entirely new graph technology of which I am the main originator. This *Graph Technology of Medical Knowledge and Healthcare Data Structuralization* was applied as a Czech national patent (application no. PV 2004-1193) in December 2004.

Using this solution the collected attributes and relations among them as well as other medical knowledge are stored in a so-called *knowledge base*. Another graph structure named *data files* is used to store the patient data itself. Both the structures are mathematically precisely described in my Ph.D. thesis or in [5].

3.1.3 User Interfaces: Within my own work I have implemented two thick user interfaces - the applications *EHRClient* and *EHRC*. The EHRClient is a user interface application that enables a simple usage of the MUDR EHR system. The EHRC is a test client application designed mainly for debugging purposes.

In regard to the fact that the development of a comfortable user interface of an electronic health record is a far too complicated issue to be solved by a single man within a reasonable time period [6], there was an extra project assigned at the Department of software engineering of Charles University in Prague, Faculty of Mathematics and Physics that I consulted. The purpose of this project was to develop a user-friendly interface to the MUDR EHR system. The result of this project is an application called *MUDRc* (*MUDR client*) [7] that enables to take advantage of the EHR MUDR system in a flexible and a comfortable way. The MUDRc enables to enter the patients' data by user-defined forms as well as directly into the data files tree. It also includes a simple tool supporting automatic structuralization of data in the form of free-text-based discharge letters (based on regular analysis techniques and results of the diploma thesis of Jiří Semecký [8]). Furthermore, the MUDRc supports the import and export of patient's data, it supports multimedia data types (e.g. images, audio- and video-attributes) and it includes a simple decision support tool. For administration purposes, the MUDRc includes various editors, e.g. the knowledge base editor, the user-rights and policy editor, the forms editor or the output templates editor.

3.1.4 MUDR EHR from the Telemedicine Point of View: A simple solution how to use the MUDR EHR system for telemedicine purposes comes forward. It is based on the implementation of specialized CGI scripts creating various outputs for user interfaces in varied forms. In such a case, the CGI script integrates most of the logic of the client application, which then has just the presentation functionality. In the frame of the thesis I considered the possibilities to implement such CGI scripts for thin clients in the form of web browsers and the Nokia 9110 communicator. In addition, I dealt with the possibilities to implement the MUDR EHR client application running in the T-Mobile MDA device [9]. The research showed that the implementation of a thin client based on CGI scripts generating HTML outputs is realistic. However, the capabilities of such a client are limited. To use such a user interface, it is convenient to fix the knowledge base. It means that we would dispose of a dynamically modifiable and extensible set of collectable features. Actually, it is not really necessary, but in case of a variable knowledge base, we get a disorganized user interface that would be hardly accepted by the physicians' public. Furthermore, we have to face the fact that the web browsers in mobile devices are much more limited than the classical HTML browsers in desktop computers. For example, the HTML browser integrated in the Nokia 9110 communicator supports neither tables nor frames. This fact forces the developer to implement an extra CGI script nearly for each device supported.

As analysing the possibilities of the EHR MUDR user interface running in the T-Mobile MDA device, I have chosen another approach. Using the Windows CE .NET or Windows XP Embedded operating systems we can transfer more of the logic and functionality back to the mobile client device. In this case it is proposed to use the potential of the .NET Compact Framework and implement the communication between the user interface of the MUDR application layer based on .NET Remoting or even more universal version based on web services, which both I have tested with a positive result. The computing power of this modern mobile device brings new capabilities compared to the thin clients based on HTML or the WAP protocol; however, we still have to conform to some limitations compared to a personal computer. The implementation of a complex mobile user interface is quite difficult and time-demanding; a developer has to conform to small displays, limited controlling possibilities, lower operational memory and computing power as well as lower communication speed.

3.2. MUDRLite Electronic Health Record

Currently, most hospitals use an electronic form of health records included in their hospital or clinical information systems. But these systems are often more suitable for the hospital management than for physicians. The health record is not structured as much as necessary, it includes a lot of free-text information and the set of collected attributes is fixed and practically impossible to be extended. Physicians gathering information for the purpose of medical studies have to often use varied proprietary solutions based on MS Access databases or MS Excel Sheets. The usage of the MUDR EHR in such cases is possible, but it may be too complicated and unavailing. Furthermore, the result may not be as user-friendly as a special application dedicated to particular user needs. Those were the main reasons why I started another research and development with the aim to create a light version of an EHR system that would provide just basic functionality and would be extendable according to special needs in a particular environment.

3.2.1 MUDRLite Architecture: The MUDRLite [10] architecture is based on 2 layers. The first one is a relational database and the second layer is a MUDRLite user interface. The database schema corresponds to the particular needs and varies therefore in different environments, in contrast to the fixed database schema in the MUDR database layer. It can be designed using standard data modelling techniques, e.g. the E-R Modelling [11].

All the visual aspects and the behaviour of the MUDRLite user interface are completely described by an XML configuration file, which is loaded by the MUDRLite Interpreter at the beginning. This configuration file specifies directly in its head section the database server that should be used. The MUDRLite Interpreter establishes a connection and asks the user to login. After that the MUDRLite Interpreter creates the user interface consisting of user-defined forms and windows.

The fact that the MUDRLite Interpreter is able to handle varied database schemas often simplifies the way of importing old data stored using different databases or files. Furthermore, this feature enables to tailor the system according to special needs of data collecting in a particular environment or for the purposes of clinical studies.

3.2.2 User Interface and Dynamic Behaviour Specification: All the visual aspects and the behaviour of the MUDRLite User Interface are described completely by an XML configuration file. This file builds the user interface as a set o defined forms with various controls placed on them. Dynamic behaviour and data manipulation are described using the so-called *MUDRLite Language* (*MLL*), those constructs are included in the configuration file as well. The power of the MLL is based mainly on the power of the SQL [12] that the MLL includes. A detailed description of the configuration file and the MLL can be found in my Ph.D. thesis or in [13].

3.2.3 Custom Components and Controls: Additional functionalities can be included by user-defined custom components. It is possible to insert visual as well as non-visual components. Using this feature it is possible to develop graphically advanced components for particular needs in a special environment or computationally advanced components to provide various supplementary potency of the EHR application.

A custom component must fulfil defined requirements to be included into the MUDRLite EHR application. These requirements specify mainly the data exchange rules and policies between a component and the MUDRLite Interpreter. Moreover, a custom component must implement the interface defined and compiled separately in the `MUDRLiteInterfaces.dll` file.

3.2.4 An Example of an Advanced Custom Component: An advanced example of a comprehensive graphical custom component is the so-called *Dental-cross* component that is intended for the application of MUDRLite EHR into the area of dental medicine. This component was ordered to be developed by an external co-operator of the EuroMISE Centre according to a detailed specification, on whose preparation I participated. The data specification of this component determines the data model of the component in the relationship to the MUDRLite Interpreter and the MUDRLite database layer. This model originates from a logical data model that I have designed in close co-operation with other colleagues from the EuroMISE Centre as well as from the Department of Stomatology of Charles University in Prague, 1st Faculty of Medicine and General University Hospital in Prague. In total, it describes 28 independent entities with more than 160 data columns; entirely described in [14] in detail.

Together with these colleagues we have further generalized this model to the so-called *Dental-Medicine Data Structuralization Technology Using a Dental Cross*. This technology was applied for a Czech national patent under the no. PV 2005-229. Furthermore, the Dental-cross component itself demonstrates relatively well the possibilities of data exchange provided by the defined interfaces.

3.2.5 MUDRLite Applicability: The testing of the MUDRLite EHR has demonstrated that this electronic health record is flexible enough and that it allows dynamical changes in the database schema requiring just small changes in the XML configuration file. The two-layer architecture of this EHR separates the user interface from the data storage and in spite of the fact that it is very simple, it is suitable in many standard cases.

The verification and evaluation of the MLL language shows that it is sufficient for many applications, which is mainly because of the power of the integrated SQL. However, some constructs expressed in the MLL language are quite ponderous and thus it is not ruled out that the MLL language will be extended in the future. Possibilities of an extension are for example in the way of including arithmetical expressions or logical conditions right into the MLL.

One of the motivations to develop electronic health record applications is to simplify the sharing and migration of medicine data that is needed among various physicians together participating in the health care of a single patient with the purpose to eliminate pointlessly repeated examinations. A way to contribute to the enhancement of health care quality is to help with overcoming of the classical free-text-based discharge letters. Nowadays, most healthcare providers use a kind of medical documentation in an electronic form of health record. However, the systems used in medical out-patient departments do not often provide sufficient possibilities of data structuralization. The real structured attributes are limited to the first name, surname and birth number of a patient in many out-patient departments of general practitioners (GP). Further structuralization lies in the form of headings of different parts of a discharge letter and the GPs are lucky to have the name and birth number automatically assigned to various documents, e.g. to an order form accompanying the patient to a specialist [15]. But from the computer scientist point of view this is not sufficient. Many physicians do not realize that they could expect more but this relates to the fact that no one has offered them anything more. We can not be surprised that physicians, who need to collect specific data (e.g. for the purpose of a clinical study), use various proprietary data storing methods that are often based on ”office-software” tools, e.g. the MS Word, MS Excel or MS Access tool. During my studies I have seen a few of technically advanced physicians who have designed and created a MS Access database themselves. However, such a database was often in the form of a single disorganized table containing many various columns breaking ”all the database normal forms”. But the fault lies not on the physicians; it lies on the computer scientists who do not support the physicians enough.

The MUDRLite EHR system, which I present as a partial result of my Ph.D. thesis, represents a relatively simple solution of creating an electronic health record tailored to special needs in a particular situation. First of all, it can be used as an advanced tool for collecting of medical data, but it does not have to be limited for this purpose. Mainly thanks to the defined interface that enables the integration of custom components, it is possible to start with a simple tool used to collect medical data and extend it step by step to an advanced EHR system according to special needs.

3.3. Telemedicine Applications on Mobile Phones

Lately, we have been undergoing a significant progress in the field of mobile telecommunication. Rarely do we meet a person who does not possess a mobile phone. The progress in the telecommunication area goes together with the progress in the field of information technologies. The computing power of microprocessors commonly used in mobile phones overtakes multiply the powers of computers controlling first flights to the

Universe. A mobile phone is not just a phone anymore; it becomes an indispensable tool providing the range of services and tools, e.g. a diary, a notebook, a calculator, an alarm clock, a dictaphone, a camera, and much more. It often enables the Internet access using a WAP or an HTML browser and an email client application, it supports data connection using GPRS or HSCSD and it enables an interconnection to a computer using a cable as well as wireless using of IrDA or Bluetooth technologies. The continuously extending potential of mobile phones was the main motivation why I have analyzed the possibilities of the usage of a mobile phone as a platform for telemedicine applications. Since the mobile phones vendors frequently implement Java support into their mobile phones, it was of concern to my Ph.D. thesis and it is detailed described in it.

3.3.1 Java in Mobile Phones: Lately, the word Java is inflected in all grammatical cases in the connection with mobile phones. The Java term often identifies an object-oriented programming language with libraries of standard classes and sets of application interfaces. This set of libraries and interfaces exists in three different editions, *Java2 Enterprise Edition* (*J2EE*), *Java2 Standard Edition* (*J2SE*), and *Java2 Micro Edition* (*J2ME*). As one would guess, J2ME is the relevant edition from the mobile phone point of view. J2ME was created with the purpose of harmonizing the Java support in small devices including not only mobile phones but also other electronic devices. Therefore, J2ME is not a complex uniform specification, yet it is divided into various configurations. A configuration specifies a basic set of libraries and device features and it is further refined by so-called profiles. Each specification [16] determines the minimal hardware configuration of a device supporting it. For the purpose of telemedicine applications on mobile phones the *Connected Limited Device Configuration* (*CLDC configuration*) is relevant. It specifies the minimum of 128 kB of a permanent memory, which content must be preserved while switched on as well as switched off (but from the custom application point of view it does not have to be writeable), and the minimum of 32 kB of the memory that must be at virtual machine disposal. The device must dispose of at least a 16 bit processor (RISC/CISC) running at least on the 16 MHz frequency.

3.3.2 Mobile Java Applicability: Although the mobile Java is used mainly as a platform for various games nowadays, the analysis shows that this smallest Java edition will find its use in the telemedicine field in the future. However, it must be admitted that a small display and just a few of input keys limiting the controlling possibilities will probably be always limiting factors for comfortable and user-friendly applications. The mobile phone manufactures are not in an easy position, the users want to get a larger and more colourful display and the keys should not be too small, but the phone itself should become smaller, lighter and more powerful. For a telemedicine J2ME application the limiting factors lie in the mobile phones' hardware as well as in the software. The size of the application is strictly limited. The possibilities regarding the CLDC 1.0 and MIDP 1.0 specifications are limited as well. In developing MIDP 1.0, the specification authors were very conservative in the functionality they chose for the base profile. The absence of any type of standard security functions, in particular, proved to be very limiting. Nevertheless, the first mobile phones supporting the MIDP 2.0 profile emerged on the Czech market. This specification enables some low-level facilities, e.g. the TCP/IP sockets or even the UDP datagrams and it enables the secure HTTPS connection as well. In addition to that, the MIDP 2.0 profile rectifies the security issue through the introduction of WAP Certificate Profile (WAPCERT) support, based on the Internet X.509 Public Key Infrastructure (PKI) Certificate and the Certificate Revocation List (CRL) Profile. The introduction of PKI functionality is utilized by MIDP 2.0 to provide secure connections and digital signatures for trusted MIDlets. Trusted applications are permitted to use APIs that would otherwise be restricted by MIDP 2.0's enhanced security model. This convinces me that using the MIDP 2.0 telemedicine applications in mobile phones become a common reality in the future.

3.3.3 A Pilot Mobile Application: While analysing the possibilities of J2ME applications, I asked myself whether it would be possible and realistic to develop a full EHR client application, e.g. a kind of MUDR user interface. Unfortunately, nowadays I have found out that this is unrealistic mainly because of mentioned limitations. However, the J2ME may be sufficient to develop some simpler telemedicine applications. Thus, as a demonstration I have formalized the – already mentioned – *1999 WHO/ISH Guidelines for the Management of Hypertension* [4] into a form of pilot J2ME application, which can be launched in most mobile phones supporting Java. A more detailed description including a simple manual how to download, install, and try in one's own mobile phone can be found in the article [17].

4. Conclusion and Summary

According to the objectives of the Ph.D. thesis I have studied the possibilities of electronic health record representation and analyzed the suitability of various data storing techniques. On the basis of the EHR system requirements, e.g. the dynamically modifiable set of collected concepts, I have picked up the threads of my diploma thesis [18], I have further extended and mathematically formalized them during the work on this Ph.D. thesis and during the research and development in the frame of my employment in the Institute of Computer Science, Academy of Sciences of the Czech Republic, the EuroMISE Centre. I have proposed a completely new Graph Technology of Medical Knowledge and Healthcare Data Structuralization. Within this research my colleagues Ing. Petr Hanzlíček, Ph.D. and Prof. RNDr. Jana Zvárová, DrSc. have supported me with valuable pieces of advice and together we submitted this technology as an application of invention with a patent request to the Czech Industrial Property Office in December 2004 (application no. PV 2004-1193). I have implemented the stated technology as a data basis of the MUDR electronic health record where I have also verified the applicability of contemporary XML-based communication trends and used XML as the basis of the MUDR API interface.

It is a difficult task to harmonize all the requirements stated for en ideal electronic health record application. Thanks to the graph data storage technology the MUDR EHR meets the requirement of a structured way of data storage combined with free text information storage possibilities, the requirement of the dynamicity of the system, mainly in the way of modifiability of the set of collectable data and the requirement on the integration of pedigree information for the purpose of a patient's family history. This technology is multilingual and it enables to associate patient's data according to the events in patients treatment or life. It includes an administrative record about all changes concerning patients' data and a record about the origin of any piece of information. It integrates multimedia data, e.g. audio and video records, images, and other unspecified binary types. Furthermore, the technology enables defining of access control policies encoded in the knowledge base, which even increases the security provided by the database systems themselves.

I participated in the leadership and consulting of a software project in the Department of Software Engineering, Faculty of Mathematics and Physics, Charles University in Prague, that resulted in an advanced user interface of the MUDR EHR. This user interface called the MUDR client (MUDRc) [7] represents a thick client application supporting data structuralization by user-defined forms and by a specialized tool enabling to use a set of user-defined regular analysis rules to structuralize data from free-text-based discharge letters. The MUDRc integrates the support of data visualization and evaluation, it enables to define various types of templates for various types of data reports, prescriptions, and other documents and additionally it includes a special statistical module retrieving descriptive statistics about the population stored in the EHR system. The interface including such a module is open and thus it is possible to add more modules in the future and extend the functionalities hereby. Furthermore, the MUDRc enables an easy way of data verification according to a user-defined set of integrity rules and it mediates consulting the health record of a particular patient with formalized medical guidelines running as decision support tools at the MUDR application layer. For the MUDR EHR I have proposed the methodology and interfaces to formalize medical guidelines, which enable to integrate them as a part of the MUDR application logic. As a pilot test I have formalized and integrated the WHO/ISH Guidelines for the Management of Hypertension [4]. The other possibilities to integrate medical guidelines formalized by the GLIF model were onward considered within the diploma thesis [19].

To extend the applicability of the MUDR EHR with the stress on support in the telemedicine area, I have further analyzed the possibilities how to implement both the thin MUDR mobile clients (HTML / WAP) and the thick MUDR mobile clients (PDA and similar devices). For the purpose of this analysis I have implemented several software components, which showed the barriers. The implementation of a complex mobile user interface is quite difficult and time-demanding; the limitations of small displays, controlling possibilities, operational memory, and computing power imply that it might be necessary to maintain a good arrangement of the user interface to fix the set of collectable features or the whole knowledge base.

Although the MUDR EHR meets practically all the requirements stated for an EHR system, the real practice showed that there are situations when there should be an effortless solution that could be tailored to special needs of the particular environment, e.g. for physicians collecting special kind of data for the purpose of a clinical study. As a response to this need I started with research and development of a lighter form of an electronic health record called MUDRLite [10], [13] that abandons the challenge to meet all requirements coming with a completely different approach. The MUDRLite EHR system is designed with the goal to provide simply the services that are needed in a small special environment and no others. The research resulted in another pilot application that simplifies the system architecture as well as the data storing principles. Furthermore, thanks to the easier conception, which enables the database schema to be user-defined, the data import possibilities are easier. However, it provides a functionality to extend the potential and capabilities of the system including advanced user-defined components that could for example enhance the security or ensure the interoperability with various other systems.

The interface was tested by integrating the Dental Cross custom component to the MUDRLite EHR system in the domain of stomatology. This component was developed by an external contractor of the EuroMISE Centre, however its data model origins in the Dental-Medicine Data Structuralization Technology Using a Dental Cross. Together with other colleagues from the EuroMISE Centre as well as from the Department of Stomatology of Charles University in Prague, 1st Faculty of Medicine and General University Hospital in Prague we have applied it for a Czech national patent under the application no. PV 2005-229.

Last but not least, I considered the possibilities of using classical mobile phones as a telemedicine applications' platform. Lately, we have been undergoing a significant progress in the field of telecommunication. A mobile phone becomes an indispensable tool providing range of services and frequently implementing support for applications in Java. Therefore, within my Ph.D. studies I have analyzed the possibilities of the Micro-Edition of the Java language with the focus on its usage for telemedicine purposes [17]. The limitations that I have described in this thesis make it impossible to develop a complex user interface of an EHR system. However, it does not mean that there is no J2ME applicability in telemedicine. As an example how to implement a telemedicine application for a mobile phone using the Java2 Micro Edition I show formalizing of the mentioned hypertension guidelines [4] as a J2ME application that everyone can test for themselves [17].

In general, I considered many electronic health record issues as well as telemedicine application issues. Some new technologies and pilot software applications were created and can be used as a motivation while developing commercial products.

The thesis was submitted with the date of defence set to October 4^{th}, 2005.

References

[1] Beale T., "Health Information Standards Manifesto.", Revision 2. 5. 2001.

[2] Bott O. J., "The Electronic Health Record – Standardization and Implementation.", In: Zywietz Ch. (ed.): *2nd OpenECG Workshop Proceedings, Integration of the ECG into the EHR & Interoperability of ECG Device Systems.* BIOSIGNA, Berlin, pp. 57-60, 2004.

[3] Mulligen E. M., Ginneken A. M., Moorman P. W., "Open Record for Care (ORCA)", electronically at http://www.eur.nl/fgg/mi/MIAnnualReports/1996/p11.html.

[4] WHO/ISH, "Guidelines for the Management of Hypertension.", In: *Journal of Hypertension.* Vol. 17, pp. 151-183, 1999.

[5] Špidlen J., Říha A. "Electronic Health Record and Telemedicine.", In: Hakl F. (ed.): *Ph.D. Conference 03.* Prague, MATFYZPRESS, ISBN: 80-86732-16-9, pp. 133-143, 2003 (in Czech).

[6] Ginneken A. M., "The Computerized Patient Record: Balancing Effort and Benefit.", In: *International Journal of Medical Informatics.* Vol. 65, pp. 97-119, 2002.

[7] Nagy M., Špidlen J., Hanzlíček P., Zvárová J.: "MUDRc – Powerfull MUDR in Medical Practice.", In: Zielinski K., Duplaga M. (eds.): *E-he@lth in Common Europe Abstracts.* Krakow, Poland, Academic Computer Centre CYFRONET UST, p. 23, 2004.

[8] Semecký J., Zvárová J., "On Regular Analysis of Medical Reports.", In: *Proceedings of NLPBA 2002.* pp. 13-16, 2002.

[9] Špidlen J., Štochl J., Semecký J., Hanzlíček P., "MUDR and Mobile Communication", In: *Medical Informatics Europe MIE 2003 Proceedings CD.* 2003.

[10] Špidlen J., Říha A. "MUDRLite - Health Record Tailored to your Needs.", In: Hakl F. (ed.): *Ph.D. Conference 04.* Prague, MATFYZPRESS, ISBN: 80-86732-30-4, pp. 153-163, 2004.

[11] Reingruber M. C., William W. G., "The Data Modeling Handbook: A Best-Practice Approach to Building Quality Data Models.", John Wiley & Sons, Inc., ISBN: 0-471-05290-6, 1994.

[12] Kriegel A., Trukhnov B. M. "SQL Bible.", Wiley Publishing, Inc., Indianapolis, ISBN: 0-7645-2584-0, 2003.

[13] Špidlen J., Hanzlíček P., Zvárová J., "MUDRLite – Health Record Tailored to your Particular Needs.", In: Duplaga M., Zielinski K., Ingram D. (eds.): *Transformation of Healthcare with Information Technologies.* Amsterdam, IOS Press, ISBN: 1-58603-438-3, ISSN: 0926-9630, pp. 202-209, 2004.

[14] Nagy M., Špidlen J., "Logický model k projektu Stoma, datový model pro MUDRLite, zubníkříž.", electronically at http://www.spidlen.cz/mudrlite/dentcrs_lm.pdf, 2004 (in Czech).

[15] Skalická H. "Právní charakter zdravotnické dokumentace a příprava lékařské zprávy v elektronické podobě.", In: Zvárová J., Přečková P. (eds.): *Informační technologie v péči o zdraví.* Prague, EuroMISE s.r.o., pp. 70-75, 2004 (in Czech).

[16] Sun Microsystems, Inc., "Introduction to Mobility Java Technology.", electronically available at http://wireless.java.sun.com/getstart.

[17] Špidlen J., "J2ME Usage in Telemedicine Applications Development for Classical Mobile Phones.", In: *Physican and technology.* 35. No.3, ČLS J. E.Purkyně, ISSN: 0301-5491, pp. 55-62, 2004 (in Czech).

[18] Špidlen J., "Database Representation of Medical Information and Guidelines.", In: *Diploma Thesis.* Prague, UK MFF, KSI, 2002 (in Czech).

[19] Kolesa P., "Analysis and Implementation of Application Server for Electronic Health Record.", In: *Diploma Thesis.* Prague, UK MFF, KSI, 2004 (in Czech).

Alternativy k evolučním optimalizačním algoritmům

doktorand:
ING. DAVID ŠTEFKA

Katedra matematiky
Fakulta jaderná a fyzikálně inženýrská ČVUT
Trojanova 13

120 00 Praha 2

david.stefka@gmail.com

školitel:
ING. RNDR. MARTIN HOLEŇA, CSC.

Ústav informatiky AV ČR
Pod Vodárenskou věží 2

182 07 Praha 8

martin@cs.cas.cz

obor studia:
Matematické inženýrství
číselné označení: 39-10-9

Chtěl bych poděkovat Ing. RNDr. Martinu Holeňovi, CSc. za odborné, pečlivé a trpělivé vedení mé práce.

Abstrakt

V tomto článku jsou popsány a porovnány některé metody pro řešení takzvané technické optimalizace, tj. hledání optima účelové funkce za situace, kdy máme k dispozici pouze velmi malý počet vyhodnocení účelové funkce a vyhodnocení účelové funkce je nákladné. Hlavní důraz je kladen na porovnání genetických algoritmů a některých dalších stochastických optimalizačních metod na základě testování na analytických funkcích a také na aproximacích funkcí z praxe.

1. Úvod

V oblasti globální optimalizace, tj. hledání globálního extrému účelové funkce, se setkáváme se dvěma základními druhy optimalizace - "modelovou" a "technickou". Jako *modelovou optimalizaci* chápeme situaci, kdy k řešení zadaného problému máme k dispozici vhodný matematický model. Tento model nám dává účelovou funkci, která je většinou analytická, nebo je alespoň algoritmicky vyčíslitelná. Vyhodnocení funkční hodnoty v zadaném bodě tedy netrvá příliš dlouho a při výpočtu si můžeme dovolit vysoký počet funkčních volání (100 000, 1 000 000 atd.). Pod modelovou optimalizaci můžeme zařadit například problém obchodního cestujícího.

Jiná situace nastává v případě *technické optimalizace*. Účelová funkce v tomto případě není analytická a nemusí být ani algoritmicky vyčíslitelná. Funkční hodnoty v různých bodech získáváme například pomocí fyzikálního měření určitých veličin nebo jako výstupy počítačových simulací. Příkladem technické optimalizace může být určení optimální koncentrace katalyzátorů v chemické reakci. Obecně pod pojmem technická optimalizace rozumíme situaci, kdy optimalizujeme účelovou funkci, jejíž vyhodnocení trvá dlouhou dobu (tedy jde o úlohy, ve kterých hraje doba vyhodnocení účelové funkce hlavní roli v době výpočtu). V případě technické optimalizace je účelová funkce většinou funkce typu "černá skříňka", což znamená, že jsme schopni pouze získat funkční hodnotu v libovolném bodě, avšak nemáme žádné dodatečné informace o účelové funkci (spojitost, diferencovatelnost, ...). Navíc vyhodnocení účelové funkce trvá dlouhou dobu a může být také finančně nákladné, nemůžeme si tedy dovolit tolik volání účelové funkce jako v případě modelové optimalizace.

Při technické optimalizaci je velmi těžké nalézt globální optimum, a proto se často omezujeme na hledání tzv. *suboptimálního řešení*, tj. řešení, které sice není globálním optimem, avšak hodnotou účelové funkce se od něj příliš neliší. Pro řešení úloh technické optimalizace se nejčastěji používají *stochastické algoritmy*, což jsou metody pro hledání optima, které pracují na základě náhodného rozhodování. Nejčastěji používanými stochastickými algoritmy jsou *genetické algoritmy*, které jsou inspirovány biologií a genetikou. Z důvodů své biologické inspirace jsou tyto metody srozumitelné i nematematikům, a kvůli tomu jsou často favorizovány na úkor ostatních stochastických optimalizačních metod. Tato práce se snaží popsat některé často používané stochastické optimalizační algoritmy vhodné pro technickou optimalizaci, včetně algoritmů genetických, a porovnat jejich chování na základě testování. Metody jsou otestovány na analytických funkcích a také na funkcích aproximovaných pomocí umělých neuronových sítí, které byly naučeny na datech z praxe, a snaží se tak simulovat technickou optimalizaci.

Významným kritériem při optimalizaci je možnost paralelizace úlohy. Optimalizační algoritmy mohou být totiž implementovány jako paralelní systém – vyhodnocujeme účelovou funkci v několika různých bodech současně. V praxi se často setkáváme se situací, kdy můžeme získat hodnotu účelové funkce v několika bodech najednou (například máme k dispozici několik nezávislých čidel měřících určitou veličinu). Proto jsou metody porovnávány v paralelní verzi, kdy se v každém kroku optimalizace provede několik vyhodnocení účelové funkce. Takovéto vyhodnocení budeme nazývat *sdružené volání účelové funkce* a počet vyhodnocení účelové funkce během jednoho sdruženého volání účelové funkce budeme nazývat *možnost paralelizace úlohy*.

2. Stochastické algoritmy

Mezi stochastické algoritmy patří algoritmy jako simulované žíhání, metoda adaptivního prohledávání, stochastická metoda větví a mezí, ale také evoluční algoritmy. Mezi evoluční algoritmy patří např. algoritmus SOMA (Self Organizing Migration Algorithm), diferenciální evoluce, optimalizace mravenčí kolonií, metoda imunologického systému, a především algoritmy genetické. Obecnými evolučními algoritmy se zde zabývat nebudeme (neboť v praxi se používají především algoritmy genetické), případný zájemce může jejich popis a odkazy na další literaturu nalézt například v [5]. Ze stochastických algoritmů popíšeme čistě náhodné prohledávání, stochastický horolezecký algoritmus, simulované žíhání a základní verzi genetického algoritmu. Popis některých dalších stochastických optimalizačních algoritmů lze nalézt například v [3].

2.1. Čistě náhodné prohledávání

Jedná se o nejjednodušší stochastický algoritmus. Je založen na náhodném výběru vzorků přípustných řešení a vybrání toho řešení, jež má optimální funkční hodnotu. V principu jde o matematickou aplikaci metody pokus-omyl. V praxi je tento algoritmus samostatně nepoužitelný, lze jej však použít k získání odhadů extremální funkční hodnoty – využitím této informace můžeme totiž mnoho algoritmů urychlit.

Při běhu algoritmu nijak nevyužíváme znalostí získaných z účelové funkce. Jedinou výhodou čistě náhodného prohledávání je, že jde o metodu vysoce paralelní. Při paralelní implementaci nám stačí de facto jediné sdružené volání účelové funkce (pokud má úloha neomezenou možnost paralelizace). Pro úlohy s velmi vysokou možností paralelizace může tedy čistě náhodné prohledávání představovat kvalitní algoritmus. V této práci jej použijeme pro porovnání jednotlivých algoritmů.

Algoritmus: Hledáme minimum funkce $f(\mathbf{x})$, kde $\forall i \in \hat{n}\ (m_i \leq x_i \leq M_i)$.

1. Náhodně vybereme přípustné řešení $\mathbf{x}$ na základě rovnoměrného rozdělení.
2. Je-li $f(\mathbf{x})$ lepší než dosud nejlepší nalezená funkční hodnota $f(\mathbf{x}^*)$, uložíme do $\mathbf{x}^*$ hodnotu $\mathbf{x}$ a uložíme si $f(\mathbf{x})$.
3. Opakujeme od kroku 1. Algoritmus ukončíme po pevném počtu iterací.

Paralelní verze metody čistě náhodného prohledávání může být implementována tak, že v každé iteraci provedeme tolik kroků neparalelního čistě náhodného prohledávání, kolik je možnost paralelizace úlohy. V každé iteraci tedy provedeme jedno sdružené volání účelové funkce.

2.2. Stochastický horolezecký algoritmus

Stochastický horolezecký algoritmus v každém kroku numericky prohledá okolí aktuálního přípustného řešení (sousedství) a přesune se do bodu ze sousedství s nejlepší hodntou účelové funkce. Sousední řešení jsou generována následovně: z aktuálního bodu se posuneme v náhodně vybrané složce o náhodnou vzdálenost tak, abychom neopustili oblast přípustných řešení. Jinou možností je náhodně generovat v krychli s hranou $2m$ (kde m je konstanta udávající maximální velikost kroku v každé složce) a těžištěm v aktuálním řešení s bodů (s ohledem na omezení stavového prostoru), a z nich vybrat řešení s nejlepší hodnotou účelové funkce.

Jde o algoritmus spíše lokálního charakteru, i když při použití druhé z výše uvedených metod generování sousedních řešení se při dostatečně velké hodnotě m zvyšuje jeho globálnost. Uvádíme jej zde proto, abychom mohli porovnat efektivitu ostatních algoritmů vzhledem k algoritmu s lokálním charakterem.

Algoritmus: Hledáme minimum funkce $f(\mathbf{x})$, kde $\forall i \in \hat{n}\ (m_i \leq x_i \leq M_i)$.

1. Inicializace – zvolíme náhodně na základě rovnoměrného rozdělení přípustné řešení $\mathbf{x^{(0)}}$. Nastavíme hodnotu konstanty s (počet generovaných sousedních bodů).
2. Iterace – pro aktuální řešení $\mathbf{x^{(k)}}$ vygenerujeme s sousedních bodů a přesuneme se do bodu $\mathbf{x^{(k+1)}}$ ze sousedství s nejlepší hodnotou účelové funkce. Po celou dobu výpočtu máme uloženo v paměti dosud nejlepší dosažené řešení $\mathbf{x}^*$. Body ze sousedství bodu $\mathbf{x^{(k)}}$ vybíráme následovně:
   (a) Vybereme náhodně složku $j \in \hat{n}$.
   (b) V j-té složce se posuneme o náhodnou vzdálenost tak, abychom neopustili interval $\langle m_j, M_j\rangle$:
   $$x_j = x_j^{(k)} + u(M_j - x_j^{(k)})\ pro\ u \in \langle 0, 1)\,,$$
   $$x_j = x_j^{(k)} + u(x_j^{(k)} - m_j)\ pro\ u \in (-1, 0),$$
   $$x_i = x_i^{(k)}\ pro\ i \neq j,$$
   kde $u \sim \mathcal{U}\,\langle -1, 1\rangle$ je náhodné číslo mezi -1 a 1 na základě rovnoměrného rozdělení.
3. Opakujeme krok 2. Po dosažení určitého počtu iterací algoritmus zastavíme s výsledkem $\mathbf{x}^*$.

Při paralelní implementaci stochastického horolezeckého algoritmu v každém kroku (tj. při každém sdruženém volání účelové funkce) vygenerujeme tolik sousedních bodů, kolik je možnost paralelizace úlohy (tedy de facto nastavíme hodnotu konstanty s rovnu možnosti paralelizace úlohy). Čím vyšší je možnost paralelizace úlohy, tím lépe prohledáme okolí aktuálního přípustného řešení.

2.3. Simulované žíhání

Princip simulovaného žíhání můžeme zjednodušeně popsat takto: Náhodně vybereme přípustné počáteční řešení $\mathbf{x} \in \mathbf{R^n}$ a nastavíme počáteční teplotu $T = 1$ (teplota je parametr určující globálnost algoritmu a během výpočtu se mění). Z bodu $\mathbf{x}$ se náhodně posuneme do bodu $\mathbf{x}'$ a určíme $f(\mathbf{x})$ a $f(\mathbf{x}')$. Je-li funkční hodnota v bodě $\mathbf{x}'$ stejná nebo lepší než v $\mathbf{x}$, přesuneme se do bodu $\mathbf{x}'$, tedy $\mathbf{x} := \mathbf{x}'$. Je-li funkční hodnota horší, můžeme se do tohoto (horšího) bodu také přesunout, ale pouze s určitou pravděpodobností. Tato pravděpodobnost je funkcí rozdílu funkčních hodnot, aktuální teploty a také funkční hodnoty v bodě $\mathbf{x}$ a s klesající teplotou klesá. Konkrétních předpisů pro výpočet této pravděpodobnosti je několik, můžeme si vybrat, který se nám pro danou úlohu lépe hodí. Po každém kroku (případně po každém zhoršujícím kroku) snížíme teplotu podle tzv. ochlazovacího plánu. Můžeme například použít tzv. simulované hašení dle vzorce $T' = T/(1 + \beta T)$. Koeficient β je malá nezáporná konstanta, $\beta < 1$. Pokud se teplota sníží pod určitou mez, tzv. zmrazení, nastavíme opět $T = 1$. Po přijmutí/odmítnutí bodu $\mathbf{x}'$ opakujeme celou proceduru výběrem nového přípustného řešení. Tím, že s určitou zmenšující se pravděpodobností přijímáme i horší řešení, donutíme funkci opustit okolí lokálního optima a přitom se stále blížit k optimu globálnímu. Při běhu simulovaného žíhání je zpočátku prohledáván globálně celý stavový prostor, později (při nižší teplotě je nižší pravděpodobnost přijmutí horšího řešení) má simulované žíhání spíše lokální charakter.

Algoritmus: Hledáme minimum funkce $f(\mathbf{x})$, kde $\forall i \in \hat{n}\ (m_i \leq x_i \leq M_i)$. Zvolili jsme konkrétní ochlazovací plán, výpočet pravděpodobnosti akceptace horšího řešení a metodu generování sousedních bodů.

1. Inicializace – zvolíme počáteční přípustné řešení $\mathbf{x^{(0)}} \in \mathbf{R^n}$ a nastavíme teplotu $T^{(0)} = 1$
2. Iterace – pro aktuální řešení $\mathbf{x^{(k)}}$ vygenerujeme jedno sousední řešení $\mathbf{x^{(k+1)}}$:
   (a) Vybereme náhodně složku $j \in \hat{n}$
   (b) V j-té složce se posuneme o náhodnou vzdálenost tak, abychom neopustili interval $\langle m_j, M_j \rangle$:
   $$x_j^{(k+1)} = x_j^{(k)} + u(M_j - x_j^{(k)})\ pro\ u \in \langle 0, 1)\,,$$
   $$x_j^{(k+1)} = x_j^{(k)} + u(x_j^{(k)} - m_j)\ pro\ u \in (-1, 0),$$
   $$x_i^{(k+1)} = x_i^{(k)}\ pro\ i \neq j,$$
   kde $u \sim \mathcal{U}\,\langle -1, 1 \rangle$ je náhodné číslo mezi -1 a 1 na základě rovnoměrného rozdělení.
3. Je-li $f(\mathbf{x}^{(k+1)}) \leq f(\mathbf{x}^{(k)})$, akceptujeme $\mathbf{x}^{(k+1)}$, teplotu nesnižujeme $T^{(k+1)} = T^{(k)}$ a opakujeme od kroku 2.
   Je-li $f(\mathbf{x}^{(k+1)}) > f(\mathbf{x}^{(k)})$, potom rozhodneme zda akceptovat řešení:
   (a) Spočítáme pravděpodobnost přijetí horšího řešení :
   $$P(\triangle, T^{(k)}) = e^{-\frac{\triangle}{T^{(k)}}}\quad ,$$
   kde $\triangle = f(\mathbf{x}^{(k+1)}) - f(\mathbf{x}^{(k)})$ a $T^{(k)}$ je teplota v k-tém kroku.
   (b) Zvolíme náhodně $p \sim \mathcal{U}\,\langle 0, 1 \rangle$.
   (c) Je-li $p \geq P(\triangle, T^{(k)})$, bod $\mathbf{x}^{(k+1)}$ neakceptujeme: $\mathbf{x}^{(k+1)} := \mathbf{x}^{(k)}$, teplotu nesnižujeme: $T^{(k+1)} = T^{(k)}$ a opakujeme od kroku 2.
   (d) Je-li $p < P(\triangle, T^{(k)})$, akceptujeme horší řešení $\mathbf{x}^{(k+1)}$ a snížíme teplotu podle ochlazovacího plánu:
   $$T^{(k+1)} = \frac{T^{(k)}}{1 + \beta T^{(k)}}\quad ,$$
   kde $\beta \in (0, 1)$ je malá konstanta, vhodné je např. $\beta = 0.1$. Klesne-li teplota pod určitou mez, např. 0.01, nastavíme opět $T^{(k+1)} = 1$.
4. Zvětšíme k o 1 a opakujeme od kroku 2. Po určitém pevném počtu iterací výpočet zastavíme.

Simulované žíhání můžeme implementovat jako paralelní algoritmus tak, že budeme prohledávat v každém kroku tolik nezávislých cest, kolik je možnost paralelizace úlohy (algoritmus tedy spustíme nezávisle tolikrát, kolik je možnost paralelizace úlohy). Jako průběžné nejlepší řešení bereme nejlepší řešení ze všech prohledávaných cest.

2.4. Genetické algoritmy

Genetické algoritmy jsou inspirovány teorií evoluce a přirozeného výběru a zákony genetiky. Tyto principy jsou aplikovány v matematice (a v mnoha jiných oborech) pro vytvoření nových, zatím nevyzkoušených postupů. Snažíme se napodobit přírodu v tom, jak přirozeně vybírá silnější jedince, a tento výběr aplikovat při optimalizaci – vytvoříme populaci jedinců, lepší jedince spolu křížíme a ty horší necháme vyhynout.

Před popisem funkce genetického algoritmu je vhodné se stručně zmínit o několika poznatcích z biologie a genetiky a s tím spojenou terminologií. Všechny živé organismy se skládají z buněk, v každé buňce je stejná sada *chromozomů*, skládajících se z molekul deoxyribonukleové kyseliny DNA. Chromozomy určují vlastnosti celého organismu a skládají se z *genů*, což jsou bloky DNA. Každý gen reprezentuje určitou vlastnost, například barvu očí. Celá genetická informace organismu se nazývá *genom*. Konkrétní nastavení genů se nazývá *genotyp*. Spolu s vnějšími vlivy po narození, jako je například výchova, určuje genotyp jedince tzv. *fenotyp*, jeho fyzické a psychické vlastnosti (barva vlasů, ale také inteligence atd.).

Při reprodukci organismů dochází ke *křížení* neboli *rekombinaci (Crossover)*. Při tomto procesu se vybírají geny rodičů a jejich kombinací se vytváří genotyp nového jedince – *potomka (Offspring)*. Může také docházet k náhodným *mutacím*, tedy změnám malé části genotypu. *Úspěšnost (Fitness)* organismu hodnotíme jako pravděpodobnost toho, že přežije do své reprodukce, nebo jako počet jeho potomků. Reprodukci si zajistí pouze některé organismy a s vyšší pravděpodobností se to podaří těm úspěšnějším.

Předchozím textem je v podstatě činnost genetických algoritmů popsána, neboť jde pouze o aplikaci tohoto postupu na problém optimalizace. Přípustné řešení v optimalizační úloze nazýváme *jedinec (Individual, Specimen)* (Pozn.: Uvažujeme, že každý jedinec se skládá pouze z jednoho chromozomu, a tedy můžeme tyto pojmy zaměňovat.). Genu odpovídá složka vektoru přípustného řešení, v aplikacích na počítači – tedy v binární reprezentaci – je však někdy výhodnější pracovat s genem jako s jedním bitem. Chromozom neboli jedinec je tedy kolekce genů, která bývá reprezentována jako bitový řetězec pevné délky. *Populací* rozumíme soubor všech jedinců, množinu jedinců vygenerovaných ve stejné iteraci nazýváme *generací*. Dále se setkáváme s pojmem *úspěšnostní funkce (Fitness Function)*, což je funkce určující zdatnost jedince, a tedy určující nějakým způsobem pravděpodobnost, že se jedinec dožije své reprodukce. V případě optimalizace bereme za úspěšnostní funkci účelovou funkci. K operacím s geny se používají tzv. *genetické operátory*, jako operátor křížení, mutace atd.

Genetický algoritmus v základní verzi pracuje tak, že z populace jedinců vybírá na základě jejich úspěšnostní funkce dvojice jedinců (rodiče), ze kterých požitím operátoru křížení vytvoří dva potomky. Lepší jedinci (tedy ti s vyšší hodnotou úspěšnostní funkce) mají větší pravděpodobnost vybrání k reprodukci, avšak všichni jedinci mají pravděpodobnost reprodukce nenulovou (tedy i nejhorší jedinec v populaci má šanci, i když malou, být vybrán k reprodukci). Každý potomek je s určitou pravděpodobností podroben mutaci, tedy náhodné změně genotypu. Poté jsou nejhorší jedinci v populaci nahrazeni těmito nově vytvořenými potomky a celý proces opakujeme.

Při použití genetických algoritmů máme možnost volby různých reprezentací jedinců pomocí chomozomů, různých selekčních schemat, a také různých operátorů křížení a mutace. Popíšeme zde genetický algoritmus v takovém tvaru, v jakém byl implementován pro testování a srovnání jednotlivých stochastických optimalizačních metod. K implementaci byl použit balík [1]. Více o genetických algoritmech a případné alternativní implementace lze nalézt například v [4].

Algoritmus: Hledáme minimum funkce $f(\mathbf{x})$, kde $\forall i \in \hat{n}\ (m_i \leq x_i \leq M_i)$. Neboť úspěšnostní funkce se většinou maximalizuje, volíme jako úspěšnostní funkci $-f(\mathbf{x})$. Neuvádíme zde přesný popis použitých genetických operátorů, neboť jej lze nalézt v [1].

1. Inicializace – vygenerujeme náhodnou populaci jedinců. Každý jedinec je reprezentován jako řetězec (reálných čísel) jednotlivých složek odpovídajícího vektoru přípustného řešení (tedy geny jsou jednotlivé složky vektoru přípustného řešení).

2. Ohodnocení – pro každého jedince vypočteme hodnotu úspěšnostní funkce.

3. Test konce – pokud jsme již dosáhli předepsaného počtu iterací, algoritmus ukončíme.

4. Vytvoření nové populace:

   (a) Selekce – pomocí selekčních schemat vybereme náhodně 2 rodiče (čím vyšší hodnota úspěšnostní funkce, tím vyšší pravděpodobnost výběru). Jako selekční schema bylo použito "normGeomSelect" z balíku [1]. Jedná se o selekci pořadím (jedinci jsou nejprve seřazeni do posloupnosti podle hodnot úspěšnostní funkce, a poté jsou vybráni dva rodiče, přičemž pravděpodobnost výběru konkrétního jedince závisí na pořadí jedince v této posloupnosti), založenou na normalizovaném geometrickém rozdělení.

(b) Křížení – pomocí operátoru křížení provedeme křížení rodičů pro vytvoření potomků. K implementaci byl použit operátor "arithXover" z balíku [1], který potomky vytváří tak, že zvolí na základě rovnoměrného rozdělení náhodné číslo $a \sim \mathcal{U}\langle 0, 1\rangle$, a poté provádí interpolaci obou rodičů:

$$potomek_1 = a \cdot rodic_1 + (1-a) \cdot rodic_2$$

$$potomek_2 = (1-a) \cdot rodic_1 + a \cdot rodic_2$$

(c) Mutace – pro každý gen každého jedince s danou pravděpodobností provedeme mutaci pomocí operátoru mutace. Jako mutační operátor byl vybrán operátor "nonUnifMutation" z balíku [1], který mění konkrétní gen na základě rozdělení, jehož rozptyl klesá s rostoucím počtem generací. Zpočátku tedy tato mutace způsobuje velké změny, později jen malé.

(d) Nahrazení – nahradíme starou populaci novou.

5. Opakování – vrátíme se k bodu 2.

Genetické algoritmy jsou již ze své podstaty paralelní. Pro paralelní implementaci tedy stačí v každé iteraci vytvořit tolik nových potomků, kolik je možnost paralelizace úlohy. Také můžeme vytvořit několik nezávislých populací, ve kterých probíhá optimalizace odděleně – tím můžeme snížit pravděpodobnost předčasné konvergence algoritmu do lokálního optima.

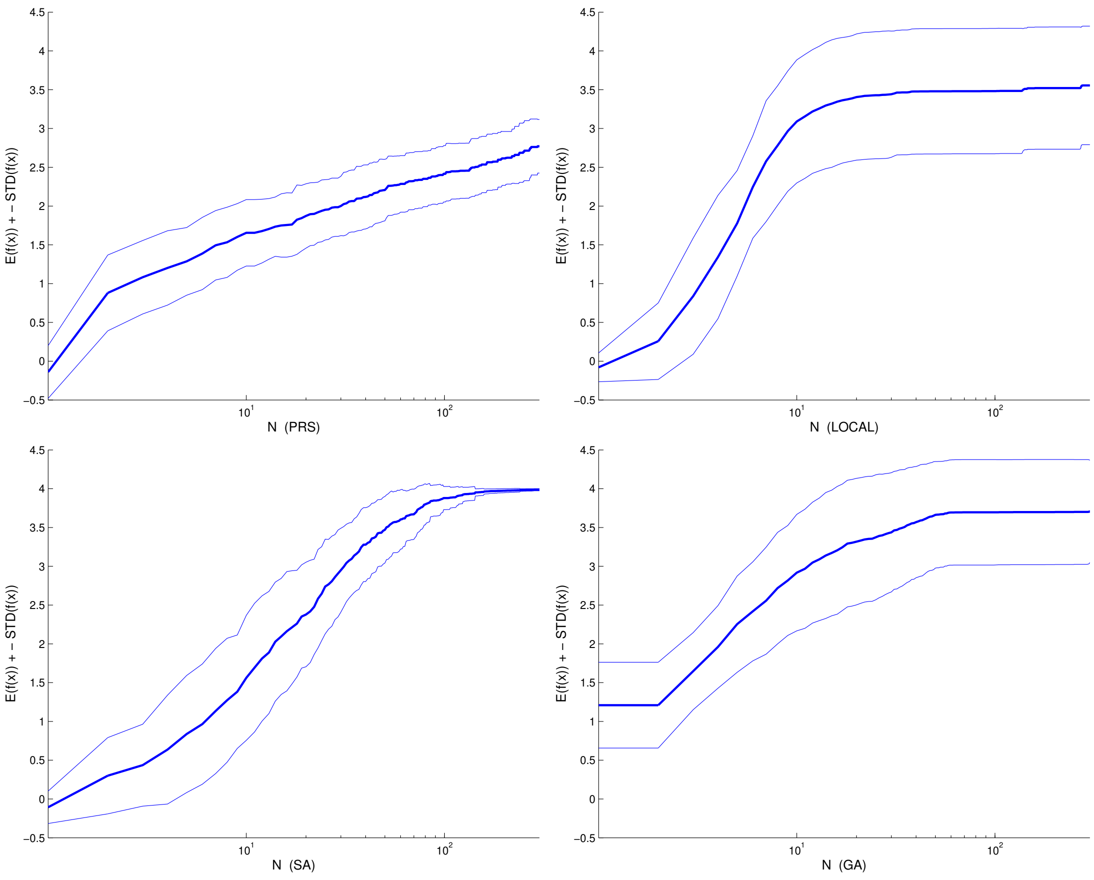
3. Testování algoritmů

Pro účely porovnání byly výše uvedené algoritmy implementovány v prostředí MATLAB. Při implementaci byl důraz kladen především na to, aby žádná metoda nebyla zvýhodněna. Metody byly otestovány na často používaných analytických testovacích funkcích (První a Druhá De Jongova funkce a Ackleyho funkce) a také na čtyřech funkcích aproximovaných pomocí umělých neuronových sítí, které byly naučeny na datech z praxe. Možnost paralelizace úlohy byla volena 8, 32 a 128. Každý algoritmus byl spuštěn 50-krát nezávisle, pokaždé bylo provedeno 500 iterací (tedy 500 sdružených volání účelové funkce). Při běhu algoritmu byla měřena střední hodnota, rozptyl, směrodatná odchylka a 0.1-, 0.5- a 0.9-kvantily funkční hodnoty v dosud nejlepším nalezeném extrému účelové funkce (přes všech 50 běhů). Tyto veličiny v dalším textu značíme $\mathbf{E}(f(x))$, $\mathbf{D}(f(x))$, $\mathbf{STD}(f(x))$, $x_{0.1}$, $x_{0.5}$ a $x_{0.9}$, N značí index iterace (počet dosavadních sdružených volání účelové funkce). Podrobný popis implementace a testování algoritmů lze nalézt v [4], kde jsou k dispozici i výsledná data z testování. Vzhledem k rozsahu tohoto textu není možné zde prezentovat všechny výsledky, proto pouze ve stručnosti shrneme výsledky testování a uvedeme grafy závislostí měřených veličin pro jednu konkrétní funkci.

Výsledky porovnání na analytických funkcích naznačují, že pro tuto třídu funkcí (tedy pro jednoduché analytické spojité funkce) není simulované žíhání vhodné, výhodnější je použít stochastický horolezecký algoritmus nebo algoritmus genetický. Čistě náhodné prohledávání dosahuje nejhorších výsledků. Pokud máme k dispozici pouze velmi malý počet sdružených volání účelové funkce a vysokou možnost paralelizaace úlohy (což odpovídá technické optimalizaci), dosahuje genetický algoritmus lepších výsledků než stochastický horolezecký algoritmus. Analytické funkce, na kterých byly algoritmy testovány, však nelze brát jako příklad technické optimalizace, neboť v praxi se setkáváme s podstatně složitějšími funkcemi. Proto jsou pro nás důležitější výsledky porovnání algoritmů na aproximacích funkcí z praxe:

Z porovnání jednotlivých metod na funkcích aproximovaných pomocí umělých neuronových sítí vychází jako nejvhodnější algoritmus simulované žíhání. Genetický algoritmus a stochastický horolezecký algoritmus dosahují ve většině případů dobrých hodnot střední hodnoty a mediánu, avšak dosahují vysokých hodnot rozptylu a v charakteristice 0.1-kvantilu jsou často horší než čistě náhodné prohledávání (je to dáno pravděpodobně jejich častou předčasnou konvergencí). Oproti tomu simulované žíhání sice konverguje pomaleji, ale dosahuje podstatně lepších hodnot 0.1-kvantilu a nižších hodnot rozptylu než ostatní algoritmy. Pokud však máme k dispozici pouze velmi malý počet sdružených volání účelové funkce (řádově desítky), ukazuje se, že u většiny testovacích funkcí je výhodnější použít genetický algoritmus nebo stochastický horolezecký algoritmus. Čistě náhodné prohledávání opět dosahuje nejhorších výsledků. Simulované žíhání se tak na základě těchto testů jeví (za předpokladu, že máme k dispozici dostatečný počet sdružených volání účelové funkce) jako nejvhodnější metoda pro technickou optimalizaci z metod prezentovaných v této práci.

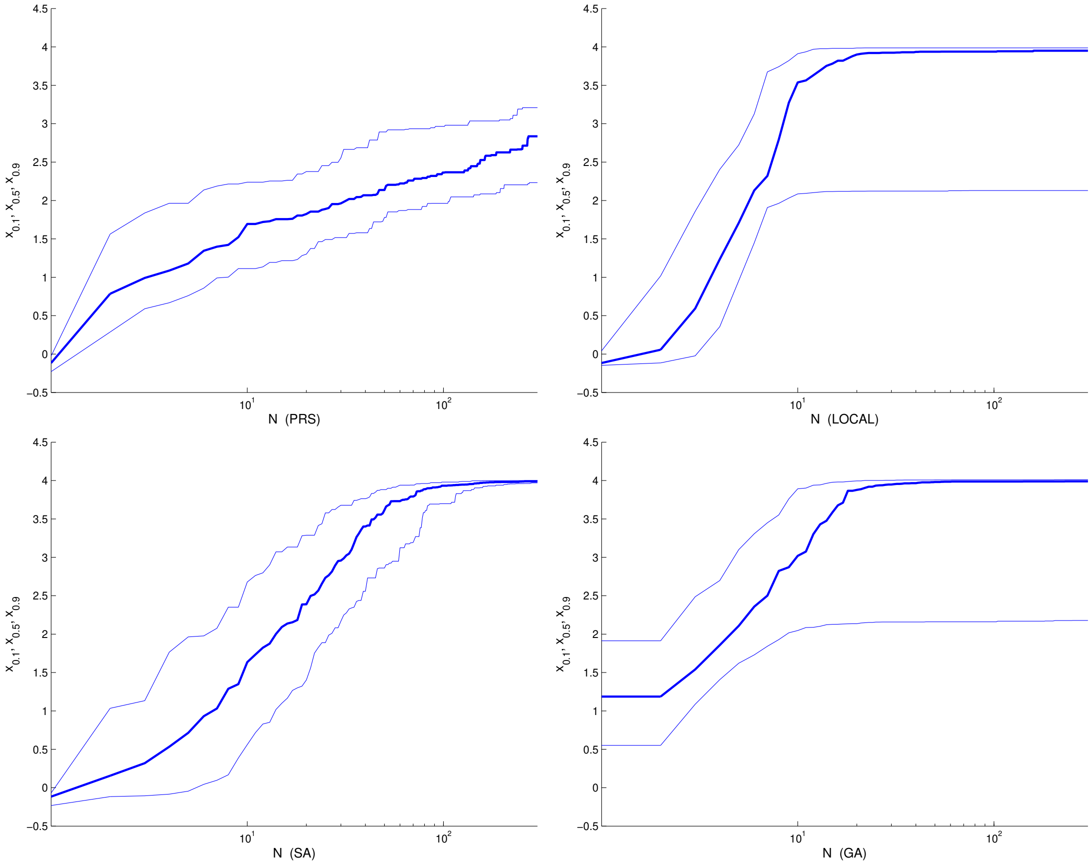
V tomto textu prezentujeme výsledky získané na jedné z aproximací funkcí z praxe, neboť je zde dobře patrný odlišný charakter jednotlivých metod. Na obrázku 1 je znázorněn graf závislostí $\mathbf{E}(f(x))$, $\mathbf{E}(f(x))+\mathbf{STD}(f(x))$ a $\mathbf{E}(f(x))-\mathbf{STD}(f(x))$ na N pro jednotlivé metody, graf závislosti $x_{0.1}$, $x_{0.5}$ a $x_{0.9}$ na N je na obrázku 2. X-ová osa grafů je logaritmická. PRS značí čistě náhodné prohledávání, LOCAL stochastický horolezecký algoritmus, SA simulované žíhání a GA genetický algoritmus. Z těchto obrázků je vidět, že simulované žíhání konverguje sice pomaleji, avšak dosahuje nižších hodnot rozptylu a lepších hodnot 0.1-kvantilu, což jej činí "spolehlivějším".



Obrázek 1: Závislost $\mathbf{E}(f(x))$ *a* $\mathbf{E}(f(x)) \pm \mathbf{STD}(f(x))$ na N pro možnost paralelizace 32

4. Shrnutí

Z výsledků testování algoritmů čistě náhodného prohledávání (jako referenčního algoritmu), stochastického horolezeckého algoritmu a simulovaného žíhání (jako zástupců tradičních stochastických optimalizačních algoritmů) a genetického algoritmu (jako zástupce evolučních algoritmů) vyplývá, že evoluční algoritmy mohou představovat kvalitní přístup k optimalizaci, není možno je však favorizovat pouze kvůli jejich snadné srozumitelnosti. Některé tradiční stochastické algoritmy mohou být totiž srovnatelné s evolučními



Obrázek 2: Závislost $x_{0.1}$, $x_{0.5}$ a $x_{0.9}$ na N pro možnost paralelizace 32

algoritmy, případně mohou být i lepší.

Pokud řešíme optimalizační úlohu, je vždy dobré nasbírat co nejvíce informací o optimalizované funkci a s ohledem na tyto skutečnosti se rozhodnout, který algoritmus k hledání optima použít. Především je dobré zjistit, jaký charakter má účelová funkce (zda je spojitá, přibližně konstantní, příbližně lineární, atd.), jaký je její definiční obor (například genetické algoritmy jsou schopny optimalizovat funkce, jejichž některé složky jsou diskrétní), má-li mnoho lokálních extrémů (pak je lepší použít globální algoritmus, např. simulované žíhání), nebo pouze jedno globální optimum (pak je lepší použít genetický algoritmus nebo nějaký lokální algoritmus, např. stochastický horolezecký), vzít v úvahu, jaká je možnost paralelizace úlohy, kolik máme k dispozici sdružených volání účelové funkce (pokud je tento počet malý, je lepší zvolit genetický algoritmus nebo stochastický horolezecký algoritmus) atd.

Jsou však i situace, kdy většinu z těchto informací nejsme schopni zjistit, a přesto musíme vybrat konkrétní algoritmus. V tomto případě je zřejmě nejlepší z algoritmů popisovaných v této práci vybrat simulované žíhání, které se jeví jako nejuniverzálnější algoritmus.

Literatura

[1] C.R. Houck, J.A. Joines, M.G. Kay, "The Genetic Algorithm Optimization Toolbox (GAOT) for Matlab v5" [online]. North Carolina State University [cit. 17.7.2005], Dostupné z WWW: <http://www.ie.ncsu.edu/mirage/GAToolBox/gaot/>.

[2] M. Obitko, "Introduction to genetic algorithms" [online] [cit. 17.7.2005]. Dostupné z WWW:

<http://cs.felk.cvut.cz/~xobitko/ga>.

[3] L. Őzdamar, M. Demirhan, "Experiments with new stochastic global optimization search techniques". *Computers and Operations Research*, 27:841–865, 2000.

[4] D. Štefka, "Alternativy k evolučním optimalizačním algoritmům", *Diplomová práce*, Katedra matematiky FJFI ČVUT, 2005. Dostupné z WWW: <http://home.tiscali.cz/davidstefka>.

[5] I. Zelinka, "Umělá inteligence v problémech globální optimalizace". Praha: BEN, 2002. ISBN 80-7300-069-5.

Zobecnění testů užívaných v metodě GUHA pro vágní data

doktorand:
ING. TOMÁŠ VONDRA

Katedra matematiky
Fakulta jaderná a fyzikálně inženýrská ČVUT
Trojanova 13
120 00 Praha 2

tv@fuzzy.cz

školitel:
ING. RNDR. MARTIN HOLEŇA, DRSC.

Ústav informatiky AV ČR
Pod Vodárenskou věží 2
182 07 Praha 8

martin@cs.cas.cz

obor studia:
Matematické inženýrství
číselné označení: X11

Abstrakt

Při rozhodování na základě obecného souboru dat se často musíme vypořádat s kombinací dvou složitých problémů, totiž s vágní neurčitostí obsaženou ve zpracovávaném souboru dat a s nemožností předem zcela jasně formulovat relevantní hypotézy. Jako vhodný model vágních dat lze využít fuzzy množiny, formulací a ověřováním relevantních hypotéz se zabývají metody tzv. explorativní analýzy. Bohužel, tyto metody, mezi které je řazena i metoda GUHA, byly v minulosti konstruovány téměř výhradně pro ostrá data (data bez vágní neurčitosti). Cílem diplomové práce tedy bylo rozšíření metody GUHA tak aby byla schopna zpracovávat i vágní data.

1. Úvod

Mnoho rozhodovacích problémů sdílí dvě charakteristiky, které ve svém důsledku znamenají výraznou komplikaci. První je neschopnost jasného vymezení malého počtu relevantních hypotéz, které mají být na daném souboru dat zkoumány. Velká část rozhodovacích problémů je totiž takové povahy že potenciálních hypotéz je velmi mnoho a nelze předpokládat že bychom na začátku "uhodli" všechny zajímavé závislosti. Metody tzv. explorativní analýzy se proto ke vstupnímu souboru dat snaží přistupovat flexibilně, tj. jen s minimálními předpoklady o datech, a hypotézy formulovat až v průběhu samotného zkoumání dat a rozhodovat zda jsou daty podporovány nebo zda jim naopak data protiřečí. Do této skupiny náleží i metoda GUHA, formulující hypotézy jako formule určité modifikace predikátového kalkulu, jejíž základy jsou shrnuty v Odstavci 2.

Druhou důležitou charakteristikou je tzv. vágnost vstupních dat, což je typ neurčitosti výrazně odlišný od neurčitosti pravděpodobnostní. Vágní neurčitost je často ilustrována pomocí výrazů běžně uživaných v řeči - "přibližně," "rychle," "vysoký" a podobně. Z nich je zřejmé že rozdíl mezi pravděpodobnostní a vágní neurčitostí lze zjednodušeně vyjádřit tak že zatímco pravděpodobnostní neurčitost spočívá v nemožnosti přesně určit jednu neznámou avšak existující hodnotu pozorované veličiny, vágní neurčitost spočívá právě v tom že veličina jedné konkrétní hodnoty nenabývá. Z tohoto důvodu se pravděpodobnostní model vágní neurčitosti jeví jako velmi nevhodný, a vyvstává tedy otázka vhodného modelu vágní neurčitosti.

Bohužel, metoda GUHA resp. její konkrétní případy jsou formulovány téměř výhradně pro data ostrá, tj. data neobsahující vágní neurčitost (obsahující pouze hodnoty "ano" a "ne", případně ještě symbol reprezentující neznámé hodnoty), a je ji tedy třeba vhodným způsobem zobecnit pro vágní data. Toto rozšíření, resp. jeho popis, je obsahem Odstavce 5.

2. Základy metody GUHA

2.1. Vstupní data a jejich interpretace

Vstupní data metody GUHA sestávají z konečné tabulky o n řádcích a k sloupcích ($n, k \in \mathbf{N}$), znázorněné na Obrázku 1, jejíž řádky reprezentují pozorované objekty $\Omega = \{\omega_1, \ldots, \omega_n\}$, a ve sloupcích jsou uloženy informace o jejich vlastnostech.

	P_1	P_2	P_3	$\ldots$	P_{k-1}	P_k
ω_1	0	1	0	$\ldots$	0	1
ω_2	1	0	0	$\ldots$	1	1
$\vdots$						$\vdots$
ω_n	1	0	0	$\ldots$	1	1

Obrázek 1: Tabulka vstupních dat

Jak je naznačeno v Tabulce 1, informace o vlastnostech P_j objektů ω_i jsou v klasické metodě GUHA kódovány pomocí hodnot 0 resp. 1, značících že objekt příslušnou vlastnost nemá resp. má, a na sloupce tedy lze pohlížet jako na unární predikáty v určité modifikaci klasického predikátového kalkulu (viz. dále). Pro každou vlastnost P_j kde $j \in \widehat{k}$ je tak množina objektů Ω vlastně rozdělena na podmnožiny Ω_0^j a Ω_1^j

$$\Omega_0^j = \{ \omega \mid P_j(\omega) = 0 \} \tag{1}$$

$$\Omega_1^j = \{ \omega \mid P_j(\omega) = 1 \} \tag{2}$$

tj. vlastně množiny objektů které danou vlastnost nemají resp. mají. Predikát P_j tedy lze popsat pomocí charakteristické funkce množiny Ω_1^j, tj. funkce $\chi_j : \Omega \to \{0, 1\}$ definované předpisem

$$\chi_j(\omega) = \begin{cases} 1 & \text{pro } \omega \in \Omega_1^j \\ 0 & \text{jinak} \end{cases} \tag{3}$$

Právě interpretace tabulky vstupních dat jako soubor predikátů $P_1, \ldots, P_k$ pro objekty $\omega_1, \ldots, \omega_n$ je pro metodu GUHA velice důležitá, neboť hypotézy jsou formulovány jako formule tzv. monadického observačního predikátového kalkulu (viz. [5]). Stejně důležité je ale vyjádření predikátů P_j pomocí podmnožin Ω_1^j resp. charakteristických funkcí χ_j, což bude využito v Odstavci 4 při hledání vhodného modelu vágních dat.

Pklad 1 (Příklad vstupní tabulky) *Uvažujme skupinu čtyř osob*

$$\Omega = \{ \textit{Alžběta}, \textit{Jiří}, \textit{Petr}, \textit{Pavel} \} \tag{4}$$

na kterých jsou v rámci průzkumu sledovány tři vlastností - predikátů (namísto zavedeného značení $P_1, \ldots, P_5$ jsou užita názorná jména):

- *Kouří*
- *PijeAlkohol*
- *MáZdravotníProblémy*

Vstupní tabulka tedy bude mít čtyři řádky a tři sloupce, a může nabývat například podoby uvedené na Obrázku 2, ze které lze vyčíst například že Petr sice kouří, ale přesto nemá zdravotní problémy a naopak že Jiří nekouří ani nepije, a přesto zdravotní problémy má.

	Kouří	PijeAlkohol	MáZdravotníProblémy
Alžběta	1	1	0
Jiří	0	0	1
Petr	1	0	1
Pavel	0	1	0

Obrázek 2: Příklad vstupních dat

2.2. Hypotézy jako logické formule

Jestliže lze na sloupce tabulky 1 pohlížet jako na predikáty jistého predikátového kalkulu, nabízí se možnost hypotézy, tj. vlatně tvrzení o závislostech mezi sloupci tabulky, formulovat jako vhodné logické formule, přičemž označení "vhodné" lze interpretovat různě.

Hlavním subjektivním kritériem je skutečnost zda formule vyjadřuje takový typ závislosti který nás zajímá - například asociační nebo implikační typ závislosti, kterými se diplomová práce zabývala. Využijeme-li predikáty zavedené v příkladu 1, mohou být příkladem formulí zachycujících asociační resp. implikační závislost formule (5) resp. (6), přičemž za jednotlivé výskyty proměnné $osoba$ lze dosazovat konkrétní hodnoty z množiny Ω.

$$\text{Kouří}(osoba) \quad \& \quad \text{MáZdravotníProblémy}(osoba) \tag{5}$$

$$\text{Kouří}(osoba)\ \&\ \text{PijeAlkohol}(osoba) \quad \Rightarrow \quad \text{MáZdravotníProblémy}(osoba) \tag{6}$$

Objektivním kritériem je například volba takových formulí, jejichž pravdivost je "svázána" s celou tabulkou vstupních dat, nikoliv pouze s jedním konkrétním řádkem. To v klasickém predikátovém kalkulu zajišťují tzv. kvantifikátory, které příslušné tvrzení vztahují k celé vstupní tabulce. Předmětem našeho dalšího zkoumání tedy budou formule ve kterých jsou všechny výskyty proměnných vázány nějakým kvantifikátorem, tzv. sentence. Příkladem sentencí klasického predikátového kalkulu mohou být například formule (7) a (8).

$$\exists\, osoba\ \ (\text{Kouří}(osoba) \quad \& \quad \text{MáZdravotníProblémy}(osoba)) \tag{7}$$

$$\forall\, osoba\ \ (\text{Kouří}(osoba) \quad \Rightarrow \quad \text{MáZdravotníProblémy}(osoba)) \tag{8}$$

Zatímco pravdivost formulí (5) a (6) závisí na konkrétní volbě osoby, pravdivost formulí (7) a (8) je prostřednictvím kvantifikátorů svázána s celou vstupní tabulkou.

Stejný princip platí i v případě monadického observačního predikátového kalkulu, avšak zatímco v klasickém predikátovém kalkulu jsou užívány pouze dva kvantifikátory, totiž kvantifikátory univerzální $\forall$ a existenční $\exists$, v observačním predikátovém kalkulu je k definici kvantifikátorů využíváno tzv. přidružených funkcí, pomocí kterých lze vyjádřit v podstatě libovolné (matematicky zformulovatelné) tvrzení o konkrétní vstupní tabulce jako celku.

Nechť nadále $\mathcal{M}^k_{\{0,1\}}$ označuje množinu všech konečných tabulek o k sloupcích obsahujících pouze hodnoty 0 a 1, q buď libovolný kvantifikátor s aritou p. Potom přidruženou funkcí Af_q kvantifikátoru q je taková funkce

$$Af_q : \mathcal{M}^p_{\{0,1\}} \rightarrow \{0,1\} \tag{9}$$

která navíc splňuje podmínky

1. Af_q je invariantní vzhledem k izomorfismu
2. je rekursivní funkcí proměnných q a M

Přitom funkční hodnota funkce na dané tabulce $\mathcal{M}^p_{\{0,1\}}$ udává pravdivost tvrzení reprezentovaného příslušným kvantifikátorem, a je jeho matematickým vyjádřením. Je zřejmé že pomocí přidružené funkce lze velice jednoduše vyjádřit kvantifikátory známé z klasického predikátového kalkulu, v následujícím příkladě jsou však zavedeny dva obecné typy závislosti - asociační a implikační - a jim odpovídající velmi triviální přidružené funkce. Složitější kvantifikátory, týkající se těchto dvou případů závislosti, jejichž zobecnění pro vágní data bylo součástí diplomové práce, jsou podrobněji rozebrány dále (a velmi detailně v [5]).

Pklad 2 (Příklady zobecněných kvantifikátorů)

- ***Implikační kvantifikátor*** - *Implikační závislost (mezi sloupci) je chápána obdobně jako v případě klasického predikátového kalkulu, tj. pracuje nad tabulkou se dvěma sloupci (je binární) a sleduje zda se v některém řádku v prvním sloupci nevysktuje hodnota* 1 *zatímco ve druhém hodnota* 0 *(neboť* $1 \rightarrow 0$ *v jistém smyslu odporuje chápání klasické implikace). Přidruženou funkci je možno definovat například jako*
$$Af_{\rightarrow}(M) = \begin{cases} 0 & \textit{v některém řádku je } 1 \rightarrow 0 \\ 1 & \textit{jinak} \end{cases} \tag{10}$$

- ***Asociační kvantifikátor*** - *Na asociační závislost (mezi sloupci) je možno pohlížet jako na situaci kdy "dostatečně převládá" shoda hodnot nad rozdílností. Označíme-li počet řádků se shodnými resp. rozdílnými hodnotami* A *resp.* B*, lze přidruženou funkci asociačního kvantifikátoru definovat například jako*
$$Af_{\approx}(M) = \begin{cases} 1 & \textit{pokud } A > B \\ 0 & \textit{jinak} \end{cases} \tag{11}$$

Poznmka 1 *Lze stanovit poměrně přirozené a názorné podmínky, které musí splňovat každý kvantifikátor testující asociační a implikační závislost (viz. [5], str. 57 - 60). Asociačních a implikačních kvantifikátorů tedy lze formulovat velké množství, právě uvedené asociační a implikační kvantifikátory jsou pouze velmi jednoduché demonstrativní příklady. Další příklady implikačních kvantifikátorů lze najít například v [1].*

Při zpracovávání tabulky M samozřejmě nejsou pomocí kvantifikátorů zpracovávány pouze jednotlivé predikáty, ale z těchto predikátů jsou (pomocí logických spojek) vytvářeny složitější formule a teprve tabulka složená z těchto formulí je zpracována zvoleným kvantifikátorem. To lze chápat také tak že každý kvantifikátor q s aritou p pracuje na "virtuální" tabulce z množiny $\mathcal{M}^p_{\{0,1\}}$, jejíž sloupce $\varphi_1, \ldots, \varphi_p$ jsou pomocí logických spojek kombinovány ze sloupců původní tabulky (totiž predikátů daného kalkulu).

Otázkou je jakým způsobem formule φ_i z predikátů $P_1, \ldots, P_k$ kombinovat - jak známo lze každou formuli přepsat jako konjunktivní resp. disjunktivní normální formu, tj. jako konjunkce disjunkcí resp. disjunkce konjunkcí predikátů a jejich negací (viz. například [6]). Tyto normální formy však mají dvě podstatné nevýhody. Zaprvé nejsou určeny jednoznačně, tj. k některým formulím existuje více než jedna konjunktivní nebo disjunktivní normální forma, a zadruhé je jejich interpretace často velmi nesnadná.

Jako velmi vhodné se naopak jeví některé poměrně jednoduché tvary, například prosté konjunkce resp. disjunkce predikátů a jejich negací, tj. formule tvarů

$$\bigwedge_{i=1}^{r} Q_{\pi(i)} \qquad\qquad \bigvee_{i=1}^{r} Q_{\pi(i)} \tag{12}$$

kde Q_j označuje buď přímo predikát P_j nebo jeho negaci, π je permutace množiny $\widehat{k}$, a $r \in \widehat{k}$. Formule těchto tvarů se velmi jednoduše sestavují a vyhodnocují, a jejich interpretace je ve srovnání s konjunktivními i disjunktivními normálními formami podstatně jednodušší. Z těchto důvodů byly v diplomové práci a související implementaci využívány právě formule těchto tvarů.

3. Kvantifikátory statistické povahy

Na predikáty P_j výchozí tabulky, resp. na hodnoty formulí φ_j z nich nakombinovaných, se lze dívat jako na realizace nějakých náhodných veličin - vzhledem k tomu že oba kvantifikátory uvažované dále jsou binární, tj. jsou definovány pro tabulky z $\mathcal{M}^2_{\{0,1\}}$, označme tyto dva sloupce tabulky M jako X a Y.

Jako přirozený způsob formulace kvantifikátorů se nabízí formulace pomocí statistiky a statistického testování hypotéz. Uvažujme vstupní tabulky $M \in \mathcal{M}^p_{\{0,1\}}$ za soubor realizací vícerozměrné náhodné (každý řádek je realizace) veličiny, hypotézu H_0 vyjadřující závislost zachycenou kvantifikátorem q a alternativní hypotézu H_1, příslušnou testovou statistiku T definovanou na množině $\mathcal{M}^p_{\{0,1\}}$ a kritickou oblast $K(\alpha)$ kde $\alpha \in (0,1)$. Potom přidruženou funkci kvantifikátoru statistické povahy můžeme pomocí statistiky T definovat například jako

$$Af_q(M) = 1 \quad \Leftrightarrow \quad T(M) \in K(\alpha) \tag{13}$$

Právě takto definované asociační a implikační kvantifikátory a jejich zobecnění byly předmětem zkoumání diplomové práce. Jejich korektní odvození a definice vyžaduje podstatně více prostoru než je k dispozici na tomto místě, uveďme však alespoň příslušné hypotézy H_0 a H_1, a interpretaci statistik. Precizní popis těchto testů lze najít například v [2] a [5], případně i v [7].

3.1. Fisherův faktoriálový test a asociační závislost

V příkladu 2 byla uvedena jedna z možných interpretací asociační závislosti jako dostatečná převaha shody nad odlišností hodnot dvou formulí. Alternativní, mírně odlišná, je interpretace využívaná při formulaci asociační závislosti na základě statistiky. V tom případě je využívána následující dvojice hypotéz

$$H_0 : X, Y \text{ jsou statisticky nezávislé} \quad \text{vs.} \quad H_1 : X, Y \text{ jsou statisticky závislé} \tag{14}$$

a toto pojetí asociační závislosti je tedy ekvivalentní závislosti statistické. Zbývá tedy odvodit vhodnou statistiku použitelnou pro testování právě uvedených hypotéz.

Pro testování obecné statistické závislosti je často užíván tzv. χ^2-test (viz. například [2]), který za předpokladu platnosti hypotézy H_0 (tj. za předpokladu nezávislosti a tedy neexistence asociační závislosti) počítá pravděpobnost že vzorek bude ve smyslu Pearsonovy χ^2 statistiky vychýlen alespoň tak jako pozorovaný vzorek. Příliš nízká pravděpodobnost nás zamítnou hypotézu nezávislosti, neboť za předpokladu nezávislosti je realizace takovéhoto vzorku velmi nepravděpodobná.

V tomto případě však jeho použití není možné neboť se jedná o test asymptotický a jako jedna z podmínek jeho užití se uvádí že žádná z četností kontingenční tabulky nesmí být menší než 5. To je však velmi obtížné zaručit, neboť nejen že často máme k dispozici jen velmi malý počet vzorků, ale při transformací pomocí řezů (viz. Odstavec 4.1) takové nízké hodnoty na krajních řezech nevyhnutelně vznikají.

Existuje však test známý jako Fisherův faktoriálový test, založený na stejném principu jako test χ^2, konstruovaný však přímo pro tabulky s nízkými četnostmi. Stejně jako test χ^2 počítá pravděpodobnost vychýlení vzorků alespoň tak jako pozorovaný vzorek, nevychází však z pearsonovy χ^2 statistiky, ale z multinomického rozdělení. Podrobnosti o Fisherově faktoriálovém testu lze najít v [2], a podrobnosti o využití pro vágní data lze najít v [7].

3.2. Binomiální test a implikační závislost

Implikační závislost lze chápat tak že implikaci příznivé případy $1 \to 1$ dostatečně převládají nad případy nepříznivými $1 \to 0$. Z pohledu statistiky tedy implikační závislost lze chápat jako vysokou podmíněnou pravděpodobnost

$$P\,(\,Y = 1 \mid X = 1\,) \overset{ozn.}{=} p \tag{15}$$

a právě tato hodnota p je předmětem binomiálního testu, jehož podrobný popis lze najít například v [2] a [7]. Tento test je ve skutečnosti formulován pro testování hypotéz

$$H_0 : p \leq \theta \quad \text{vs.} \quad H_1 : p > \theta \tag{16}$$

kde $\theta \in (0, 1)$ je zvolený práh, jehož přesná hodnota však na začátku testování není k dispozici. Tento problém lze do jisté míry obejít přeformulováním hypotéz do vágního tvaru, tj. do podoby

$$H_0 : p \text{ je malá} \quad \text{vs.} \quad H_1 : p \text{ je velká} \tag{17}$$

při jejichž testování je zkombinován klasický binomiální test s vágním pojmem "velký" resp. negací pojmu "malý," a to následujícím způsobem. Nejdříve je nalezena taková hodnota θ pro kterou klasický binomiální test přechází od zamítání k přijímání (a naopak), a stupeň příslušnosti tohoto pojmu k vágnímu pojmu "velký" resp. negaci pojmu "malý" je považován za výsledek binomiálníhi testu fuzzy hypotéz. Podrobnou definici binomiálního testu pro fuzzy hypotézy lze najít v [1] a v [7].

4. Vágnost dat

Jak již bylo zmíněno, vágní neurčitost je velmi odlišná od neurčitosti pravděpodobnostní. Pravděpodobnostní neurčitost je obvykle spojována se situacemi kdy sledovaná veličina nabývá právě jedné hodnoty, kterou však není možno určit - pěkným příkladem je hod kostkou, kdy sice víme že v příštím hodu padne právě jedno celé číslo od 1 do 6, předem však nejsme schopni určit které to bude. Pokud padne například hodnota 3, nemůže současně padnout také hodnota 4 a podobně - vždy se realizuje pouze jedna jediná z možných hodnot.

Povaha vágní neurčitosti je však zcela odlišná - uvažujme například množinu reálných čísel a vágní pojem "přibližně 5." V tomto případě neexistuje jediná hodnota která by byla "přibližně 5" ale k tomuto pojmu více či méně přísluší všechny hodnoty - je pouze otázka do jaké míry. Na rozdíl od pravděpodobnostní neurčitosti se tedy nerealizuje pouze jedna jediná hodnota, ale všechna reálná čísla jsou "přibližně 5." Například hodnota 4 bude "přibližně 5" spíše než 3, ale méně než $4,5$.

4.1. Reprezentace vágních dat

Vyvstává tedy otázka jak vágnost modelovat. Nejjednodušší je zřejmě rozdělení množiny reálných čísel na podmnožiny A a B, přičemž A reprezentuje hodnoty které jsou "přibližně 5" a množina B hodnoty které "přibližně 5" nejsou. To je zřejmě intervalový model, plně popsatelný charakteristickou funkcí $\chi_A : \mathbb{R} \rightarrow \{0, 1\}$ množiny A, tj. funkcí

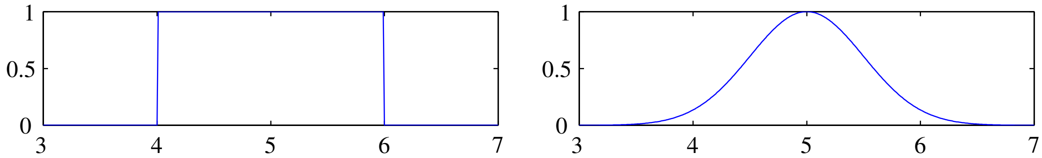
$$\chi_A(x) = \begin{cases} 1 & \text{pro } x \in A \\ 0 & \text{jinak} \end{cases} \tag{18}$$

Tento model má však jeden podstatný nedostatek - jeho chování totiž neodpovídá přirozenému chování vágních dat. Uvažujme pro názornost opět vágní pojem "přibližně 5." Při použití intervalového modelu se zcela ztrácí informace o tom jak moc je daná hodnota "přibližně 5", neboť každá hodnota buď patří do množiny A nebo nepatří. Stejně nežádoucí je však chování na hranicích množiny A. Uvažujme například $A = [4, 6]$ - proč by hodnota 4 měla být "přibližně 5" a hodnota $4 - \epsilon$ nikoliv? To zcela jasně odporuje přirozenému chápání pojmu "přibližně 5."

Oba právě uvedené důvody naznačují že intervalový model není pro vágní data vhodný, ačkoliv velmi dobře vyhovuje pro popis pravděpodobnostní neurčitosti. Důvodem nevhodnosti je právě neschopnost popsat "stupňovitou" povahu vágní neurčitosti, která je ale její nejdůležitější charakteristikou. To je ale současně návod jak model upravit aby vágní neurčitosti lépe odpovídal - stačí přejít od klasického intervalového modelu k tzv. fuzzy množinám, tj. přejít od charakteristické funkce χ_A k obecnějšímu stupni příslušnosti (19), což je ilustrováno na Obrázku 3.

$$\mu_A : \mathbb{R} \rightarrow [0, 1] \tag{19}$$

Ačkoliv volba konkrétního tvaru funkce μ_A musí odpovídat přirozenému chápání daného vágního pojmu, jedná se o velmi subjektivní otázku. Další podrobnosti o rozdílech mezi vágní a pravděpodobnostní neurčitostí, jejich kombinacích a volbě vhodného tvaru fuzzy množiny lze najít například v [4] a [3], a je jim věnována také celá první kapitola mé diplomové práce.



(a) intervalový model pojmu “přibližně 5” (b) fuzzy model pojmu “přibližně 5”

Obrázek 3: Přechod od χ_A k μ_A

Vstupní tabulka 1 vlastně odpovídá výše popsanému intervalovému modelu, neboť predikáty P_i vymezují podmnožiny Ω_i objektů které mají příslušnou vlastnost, a predikáty (3) tedy vlastně odpovídají přímo charakteristickým funkcím (18) těchto množin. Přejdeme-li k vágním datům popsaným fuzzy množinami, přejdeme vlastně od klasických predikátů P_i k fuzzy predikátům

$$\widetilde{P}_i : \Omega \rightarrow [0,1] \tag{20}$$

Bezprostředním důsledkem přechodu k fuzzy predikátům je skutečnost že Tabulka 1 vstupních dat již nebude obecně obsahovat výhradně hodnoty 0 a 1, ale libovolné hodnoty z intervalu $[0,1]$. Jaký vliv má toto zobecnění na metodu GUHA?

5. Modifikace metody GUHA pro vágní data

Ačkoliv pro manipulaci s vágními daty rerezentovanými fuzzy množinami máme k dispozici velmi mocný nástroj - fuzzy logiku, přechod k vágním datům se nutně projeví již při sestavování hypotéz. Jak již bylo uvedeno, jsou hypotézy formulovány jako sentence observačního predikátového kalkulu, tj. jsou to uzavřené formule a jejich důležitou součástí tedy nutně musí být kvantifikátor. Kvantifikátory zavedené například v [5] jsou však definovány výhradně pro vstupní tabulky z $\mathcal{M}^p_{\{0,1\}}$, nikoliv pro vágní tabulky $\mathcal{M}^p_{[0,1]}$.

Je tedy třeba zavést tzv. fuzzy zobecněné kvantifikátory, pracující na tabulkách z $\mathcal{M}^p_{\{0,1\}}$, a odpovídající pojem fuzzy přidružené funkce

$$FAf_q : \mathcal{M}^p_{\{0,1\}} \rightarrow [0,1] \tag{21}$$

kde q je fuzzy zobecněný kvantifikátor arity p.

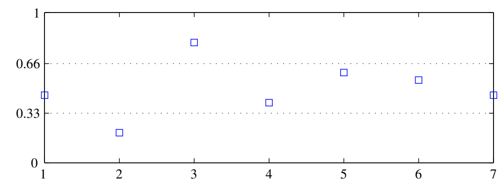
Existují v zásadě dva “extrémní” způsoby, kterými se lze s touto definicí vypořádat. První možností je zapomenout na všechny již zavedené kvantifikátory a vytvořit kvantifikátory zcela nové, čímž ale víceméně zahodíme mnohdy velký kus práce. Druhou možností je pominout vágnost, tj. tabulku transformovat řezem na “ostrou” tabulku (například všechny hodnoty nad 0.6 přepíšeme na 1 a zbytek na 0) a následně aplikovat některý z již známých kvantifikátorů, přičemž ale ztrácíme velmi podstatnou charakteristiku dat - vágnost. Ani jeden z těchto způsobů tedy není příliš vhodný.

Naštěstí existuje rozumný kompromis mezi oběma postupy, při kterém je sice využita transformace na data ostrá a na tato data jsou následně aplikovány již definované zobecněné kvantifikátory, ale transformace je prováděna takovým způsobem že velká část informace o vágnosti je zachována a negativní dopady uvedené v předchozím odstavci jsou velmi dobře vyváženy.

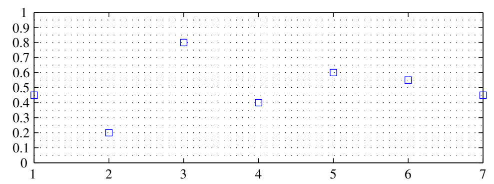
Buď μ libovolná fuzzy podmnožina množiny Ω, reprezentovaná stupněm příslušnosti (19). Potom řezem fuzzy množinou na hladině $\delta \in [0,1]$ rozumíme podmnožinu μ_δ zavednou vztahem (22)

$$\mu_\delta = \{ \ \omega \ : \ \mu(\omega) \geq \alpha \ \} \tag{22}$$

tj. podmnožinu množiny Ω do které náleží právě ty prvky ω jejichž stupeň příslušnosti k fuzzy množině μ je alespoň δ. Řezem je vlastně množina Ω rozdělena na dvě klasické podmnožiny dle stupně příslušnosti



(a) nízký počet řezů



(b) vysoký počet řezů

Obrázek 4: Potíže s skvidistantními řezy

k fuzzy množině μ, a představuje tedy jakoby postup inverzní k přechodu od intervalového modelu k modelu pomocí fuzzy množin (viz. Odstavec 4.1).

Poznmka 2 *Řez lze samozřejmě definovat také "ostře", tj. vztahem (2), vliv této modifikace na princip a fungování transformace je však minimální (viz. [3]).*

$$\mu_\delta = \{ \omega \ : \ \mu(\omega) > \alpha \}$$

Buď M libovolná vágní vstupní tabulka z $\mathcal{M}^p_{[0,1]}$, a nechť $\delta \in [0,1]$. Všechny fuzzy predikáty P_i tvořící tabulku M vlastně popisují nějakou fuzzy množinu (jsou jejím stupněm příslušnosti), a jejich řezy na hladině δ označme P_i^δ. Řez M_δ tabulky M na hladině δ definujme jako tabulku složenou z řezů predikátů P_i^δ na stejné hladině δ. Řez tabulky M na zvolené hladině δ tedy vzniká tak že všechny hodnoty ostře menší než δ jsou přepsány na hodnotu 0 a všechny ostatní hodnoty na hodnotu 1.

Pokud bychom prováděli jeden jediný řez, znamenalo by to vlastně prostou transformaci na ostrá data, tj. druhou možnost definice zobecněných kvantifikátorů, jejíž nevýhody byly popsány výše. Je tedy třeba provádět více řezů na vhodně zvolených hladinách, tvořících vlastně posloupnost Δ (viz. Definice 1), na každém řezu aplikovat příslušný kvantifikátor resp. jemu odpovídající statistický test, a takto získané výsledky testů a hladiny řezů zkombinovat do jedné hodnoty.

Definice 1 (Posloupnost řezů Δ) *Nechť $\Delta = (\delta_j)_{j=1}^m$ je konečná ostře rostoucí posloupnost, pro kterou $\delta_j \in [0,1]$. Potom Δ je nazývána* ***posloupnost řezů****.*

Poznmka 3 *Posloupnost řezů Δ vlastně dělí interval $[0,1]$ na $m+1$ disjunktních intervalů D_j, kde*

$$\begin{aligned} D_1 &= [0, \delta_1] \\ D_j &= (\delta_{j-1}, \delta_j) \quad j \in \widehat{m} \setminus \{1\} \\ D_{m+1} &= (\delta_m, 1] \end{aligned} \tag{23}$$

Význam tohoto dělení bude zřejmý z principu kombinace mezivýsledků v Odstavci 5.

Otázkou však i nadále zůstává jak hladiny testů volit. Zvolíme-li jich příliš málo nebo pokud je umístíme nevhodně, ztratíme příliš velkou část informace o vzájemném chování funkcí, což je ilustrováno na Obrázku 4(a). Naopak zvolíme-li příliš vysoký počet řezů, což je znázorněno na Obrázku 4(b), zachytíme obvykle velmi dobře informace o vzájemném chování funkcí, současně však vzroste také výpočetní náročnost (neboť na každém řezu je aplikován statistický test).

Jako optimální se tedy jeví vložit do posloupnosti Δ všechny hodnoty z tabulky vstupních dat, neboť při takové volbě řezů bude zachyceno maximum informací o vzájemném chování a současně bude prováděn minimální počet statistických testů.

Statistické testy (uvedeny byly například test Fisherův a binomiální) však nejsou prováděny až do posledního kroku, tj. rozhodnutí "ano" či "ne," ale za výsledek testu je brána hodnota statistiky - v uvedených příkladech se jedná o pravděpodobnosti určitého jevu (viz. odstavce 3.1 a 3.2 o těchto testech), kterou lze interpretovat jako stupeň příslušnosti vágních pojmů "tabulka je asociačně závislá" resp. "tabulka je implikačně závislá."

Výsledkem transformace a aplikace testů na jednotlivých hladinách je tedy tabulka dvojic hladiny δ a výsledku statistiky T_δ na této hladině, jejíž příklad je uveden na obrázku 5.

δ	0.10	0.21	$0.34 \cdots$	0.73	0.91
T_δ	0.71	0.34	$0.15 \cdots$	0.54	0.13

Obrázek 5: Příklad tabulky výsledků na řezech

5.1. Fuzzy integrály

Účelem fuzzy integrálů, jen velmi stručně popsaných v této kapilole, je kombinace výsledků testů na jednotlivých řezech do jediné hodnoty, reprezentující "sílu" závislosti. Před definicí fuzzy integrálu je třeba zavést několik následujících pojmů - fuzzy míra, t-konorma, pseudo-rozdíl a jednoduchá funkce.

Definice 2 (Fuzzy míra) *Buď X libovolná neprázdná množina a $\mathcal{X}$ takový systém jejích podmnožin, pro který platí*

1. $\emptyset \in \mathcal{X}$, $X \in \mathcal{X}$
2. *$\mathcal{X}$ je uzavřená vzhledem ke sjednocení monotonně rostoucích posloupností*

Potom fuzzy mírou na množině X nazýváme takovou funkci $\mu : \mathcal{X} \to [0, 1]$, splňující podmínky

- $\mu(\emptyset) = 0$ *a* $\mu(X) = 1$
- $A, B \in \mathcal{X}$ *a* $\subseteq B \;\Rightarrow\; \mu(A) \leq \mu(B)$
- *je monotonně spojitá*

Definice 3 (Jednoduchá funkce) *Funkce $f : X \to [0, 1]$ se nazývá jednoduchá, pokud pro všechna $x \in X$ platí*

$$f(x) = \sum_{i=1}^{n} a_i \mathbf{1}_{D_i}(x) \tag{24}$$

kde $a_i \in [0, 1]$, D_i je systém disjunktních podmnožin X s charakteristickými funkcemi 1_{D_i}.

Definice 4 (t-konorma) *T-konormou nazýváme funkci $\triangle : [0, 1] \times [0, 1] \to [0, 1]$ splňující následující podmínky:*

1. $1 \triangle 1 = 1$, $0 \triangle x = x \triangle 0 = x$ *(korespondence s Archimedeovskou logikou)*
2. $(\forall x, y \in [0, 1]) \, (x \triangle y = y \triangle x)$ *(symetrie)*
3. $(\forall x, y, z \in [0, 1]) \, (x \triangle (y \triangle z) = (x \triangle y) \triangle z)$ *(asociativita)*
4. $(\forall x, x', y, y' \in [0, 1]) \, (x \leq y, x' \leq y) \, (x \triangle x' \leq y \triangle y')$ *(monotónnost v obou parametrech)*

Poznmka 4 *T-konorem je samozřejmě nekonečně mnoho, ale tři nejčastěji užívané t-konormy jsou*

$$x \mathbin{\widehat{+}} y = \min(x+y,1) \tag{25}$$
$$x \vee y = \max(x,y) \tag{26}$$
$$x \oslash y = x+y-x\cdot y \tag{27}$$

Všechny tyto tři t-konormy jsou spojité, t-konormy Łukasiewiczova a Produktová jsou navíc Archimedeovské. T-konormy jsou v rámci fuzzy logiky využívány pro vyhodnocování logické spojky "nebo" a právě uvedené podmínky tedy zajišťují v jistém smyslu přirozené chování. Další informace o t-konormách lze najít například v [4].

Definice 5 (Pseudo-rozdíl) *Buď $\triangle$ libovolná t-konorma, potom tzv. pseudo-rozdíl $-_\triangle$ je pro každé $x, y \in [0,1]$ dán předpisem*

$$x -_\triangle y = \inf_{z\in[0,1]} \{z \,:\, x \le y \triangle z\} \tag{28}$$

Lemma 1 *Nechť $\triangle$ je libovolná spojitá t-konorma, a $-_\triangle$ příslušný pseudo-rozdíl. Potom pro všechna $x, y \in [0,1]$, taková že $x \ge y$, platí*

$$(x -_\triangle y) \triangle y = x \tag{29}$$

Pklad 3 (Pseudo-rozdíly pro běžné t-konormy) *Pseudo-rozdíly pro tři nejběžnější t-konormy (25), (26) a (27) jsou dány následujícími předpisy*

$$x -_{\widehat{+}} y = \begin{cases} x-y & x \ge y \\ 0 & x < y \end{cases} \tag{30}$$
$$x -_\vee y = \begin{cases} x & x \ge y \\ 0 & x < y \end{cases} \tag{31}$$
$$x -_\oslash y = \begin{cases} 0 & x \le y \\ \frac{x-y}{1-y} & x > y \end{cases} \tag{32}$$

Definice 6 (Integrál vzhledem k fuzzy míře) *Buď $(X, \mathcal{X}, \mu)$ fuzzy měřitelný prostor, nechť $\mathcal{F} = (\triangle, \perp, \underline{\perp}, \diamond)$ je systém tří t-konorem které jsou buď Archimedeovské nebo $\vee$, a $\diamond$ nechť je "produktová" operace, tj. nechť*

$$\diamond : [0,1] \times [0,1] \to [0,1] \tag{33}$$

Potom pro jednoduchou měřitelnou funkci $f : X \to [0,1]$ je ***integrál vzhledem k fuzzy míře*** *funkce f založený na systému $\mathcal{F}$ vzhledem k fuzzy míře μ definován jako*

$$(\mathcal{F}) \int f \diamond d\mu = \underline{\perp}_{i=1}^{n} ((a_i -_\triangle a_{i-1}) \diamond \mu(A_i)) \tag{34}$$

Fuzzy integrál je tedy plně zadán systémem $\mathcal{F}$, přičemž v diplomové práci byly podrobněji rozebírány tři integrály, zadané systémy

$$\mathcal{C} = (\widehat{+}, \widehat{+}, \widehat{+}, *) \tag{35}$$
$$\mathcal{S} = (\vee, \vee, \vee, \wedge) \tag{36}$$
$$\mathcal{P} = (\oslash, \oslash, \oslash, *) \tag{37}$$

tj. integrály Choquetův (systém $\mathcal{C}$), integrál Sugenův (systém $\mathcal{S}$) a integrál produktový (systém $\mathcal{P}$), a související úzce s jednotlivými t-konormami uvedenými v poznámce 4.

Interpretace těchto integrálů, pocházejících (s výjimkou integrálu produktového) původně z různých oborů, je poměrně komplikovaná otázka, jejíž poměrně rozsáhlý a důkladný rozbor lze najít například v [4]. Z uvedených definic a poznámek však vyplývá důležitý poznatek, a totiž že spíše než zobecněním Lebesgueova integrálu jsou fuzzy integrály agregací hodnot, závisející na zvolených t-konormách a produktové funkci.

Právě této interpretace lze s úspěchem využít pro kombinaci výsledků testů na jednotlivých hladinách řezů, ilustrovaných Tabulkou 5. Tyto výsledky lze totiž zapsat jako jednoduchou funkci, kterou lze po volbě vhodného systému $\mathcal{F}$ integrovat, tj. převést na jedinou hodnotu.

Literatura

[1] Martin Holeňa, "Fuzzy hypotheses for GUHA implications", *Fuzzy sets and systems 98*, pp. 101–125, 1998.

[2] Jiří Anděl, "Matematická statistika", *SNTL*, 1985.

[3] Rudolf Kruse, "Statistics with vague data", *Kluwer Academic Publishers*, 1987.

[4] Michel Grabisch, Hung T. Nguyen, Elbert A. Walker "Fundamentals of uncertainity calculi with applications to fuzzy inference", *Kluwer Academic Publishers*, 1995.

[5] Tomáš Hájek, Tomáš Havránek "Mechanizing Hypothesis Formation", *Springer-Verlag*, 1978.

[6] Vítězslav Švejdar "Logika - neúplnost, složitost, nutnost", *Academia*, 2002.

[7] Tomáš Vondra "Zobecnění testů užívaných v metodě GUHA pro vágní data", *diplomová práce FJFI ČVUT*, 2005.

Analýza vybraných matematických modelů pro tvorbu zpětné vazby v e-Learningu

doktorand:
ING. DANA VYNIKAROVÁ
Katedra informačního inženýrství
PEF ČZU
Kamýcká 129
165 21, Praha 6
vynikarova@euromise.cz

školitel:
PROF. RNDR. JIŘÍ VANÍČEK, CSC.
Katedra informačního inženýrství
PEF ČZU
Kamýcká 129
165 21, Praha 6
vanicek@pef.czu.cz

obor studia:
Informační management
číselné označení: 6209V

Abstrakt

Nezbytnou součástí každého elektronického výukového kurzu jsou zpětnovazební prvky. Zpětná vazba je soubor kontrolních prvků (otázky, úkoly, testy, apod.), pomocí kterých lektor sleduje práci studenta v kurzu a hodnotí jej. Dosažené výsledky jsou využitelné také samotnými studenty jako zpětná vazba jejich studia. Článek se snaží nastínit, jakým způsobem a prostředky namodelovat zpětnou vazbu, aby účelně plnila svoji funkci.

Klíčová slova: Elektronická výuka, zpětnovazební prvky, matematické modely, modelování zpětné vazby. ontologie

1. ÚVOD

Je zřejmé, že je velmi náročné vyvinout kvalitní a v praxi použitelný elektronický výukový systém, a to jak z hlediska technického provedení, tak i s ohledem na hledisko pedagogické. Je nutné, aby systém zahrnoval obě tyto roviny. K tomu, aby daný výukový systém byl kvalitní a přínosný, a to jak pro studenta, tak i pro pedagoga, musí také kromě základních funkcí (tj. výuky, testování, komunikace, konzultace, atd.) obsahovat účinné zpětnovazební prvky, pomocí kterých je možné studenta hodnotit a určitým způsobem korigovat jeho průchod výukovým systémem. Je proto nezbytné analyzovat, jak vhodným způsobem konstruovat a aplikovat do elektronického výukového systému funkční a kvalitní zpětnovazební prvky.

2. CÍL A METODIKA

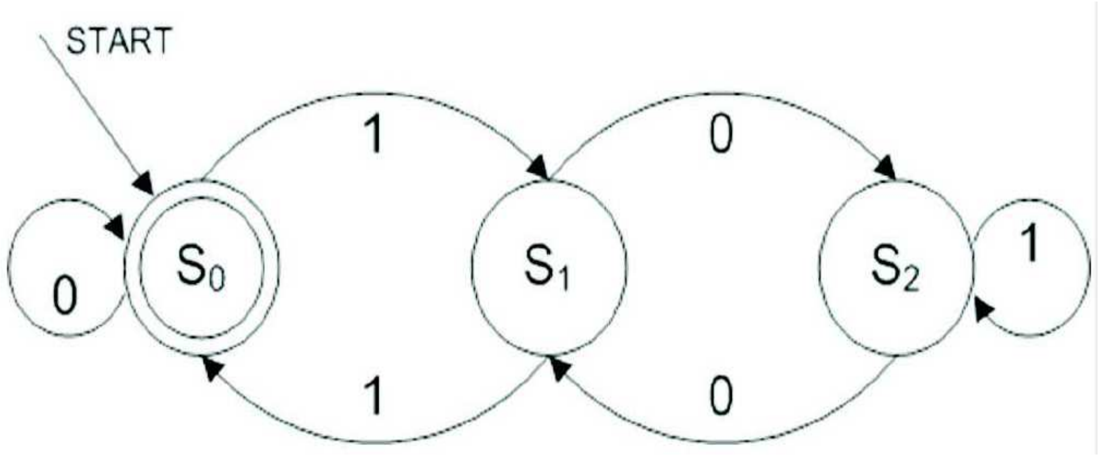
V matematické teorii výpočtu se pro popis průběhu výpočetního procesu využívá řada formálních modelů. Vzniká přirozená otázka, zda by nebylo možné tyto modely, po jejich případné modifikaci, využít i v elektronické výuce, a to pro modelování procesu průchodu studenta výukovým kurzem a využít je jak pro jeho navigaci, tak i pro zprostředkování zpráv uživatelům (tj. student, lektor, tvůrce kurzu, apod.). Z tohoto důvodu je cílem této práce vytipovat ze stávajících matematických modelů ten, který je nejvhodnější pro konstrukci zpětné vazby v elektronickém výukovém systému.

3. METODIKA

V následující části této práce se stručně zmíním o nejčastěji užívaných matematických modelech a pokusím se o jejich stručné zhodnocení z hlediska vhodnosti použití pro konstrukci zpětné vazby.

3.1. Konečný automat

Konečný automat (též FSM z anglického finite state machine) je teoretický výpočetní model používaný v informatice pro studium vyčíslitelnosti a obecně formálních jazyků. Popisuje velice jednoduchý počítač, který může být v jednom z několika stavů, mezi kterými přechází na základě symbolů, které čte ze vstupu. Množina stavů je konečná (odtud název), konečný automat nemá žádnou další paměť kromě informace o aktuálním stavu. Konečný automat je velice jednoduchý výpočetní model, dokáže rozpoznávat pouze regulární jazyky. Konečné automaty se používají pro zpracování regulárních výrazů, např. jako součást lexikálního analyzátoru v překladačích.



Obrázek 1: *Znázornění konečného automatu se stavy S_0, S_1 a S_2*

Princip činnosti konečného automatu

Na počátku se automat nachází v definovaném počátečním stavu. Dále v každém kroku přečte jeden symbol ze vstupu a přejde do stavu, který je dán hodnotou, která v přechodové tabulce odpovídá aktuálnímu stavu a přečtenému symbolu. Poté pokračuje čtením dalšího symbolu ze vstupu, dalším přechodem podle přechodové tabulky atd., až do přečtení posledního znaku zpracovávaného slova. Skončí-li automat v některém z cílových stavů, je slovo přijato. Pokud skončí mimo množinu cílových stavů, je slovo odmítnuto.

Množina všech řetězců, které konečný automat přijme, tvoří regulární jazyk. Dle [1] lze dokázat, že regulární jazyky jsou právě ty jazyky, které lze generovat pomocí tak zvaných regulárních gramatik, to je gramatik Chomského třídy 3. Právě tak lze dokázat, že regulární jazyky jsou právě ty jazyky jejichž slova lze popsat pomocí tak zvaných regulárních výrazů.

3.2. Zásobníkový automat

Zásobníkový automat (PDA z anglického *pushdown automaton*) je teoretický výpočetní model používaný v informatice pro studium vyčíslitelnosti a obecně formálních jazyků. Je to jednosměrný nedeterministický automat, který má pomocnou, potenciálně omezenou paměť organizovanou jako zásobník (tedy s přístupem LIFO, pouze na vrchol zásobníku).

Zásobníkový automat se v podstatě skládá z konečného automatu, který má navíc k dispozici potenciálně neomezenou paměť ve formě zásobníku. Obsah tohoto zásobníku ovlivňuje činnost automatu tím, že vstupuje jako jeden z parametrů do přechodové funkce.

Síla modelu zásobníkového automatu

Nejdůležitější částí zásobníkového automatu je jeho paměť (zásobník). Samotný konečný automat bez zásobníku dokáže rozpoznávat pouze regulární jazyky. "Vnitřní" konečný automat může být velice jednoduchý, dokonce s jediným stavem - důležitější částí je zásobník, který umožňuje automatu rozpoznávat bezkontextové jazyky, tedy jazyky Chomského třídy 2.

Jelikož se tedy přidáním zásobníku rozšíří třída jazyků, které automat umí rozpoznat, nabízí se otázka, zda by se téhož nedosáhlo přidáním dalšího zásobníku. A skutečně, zásobníkový automat se dvěma zásobníky má výpočetní sílu ekvivalentní Turingovu stroji, neboť jedním zásobníkem může emulovat část pásky vlevo od polohy hlavy, druhým zásobníkem pak část pásky vpravo od hlavy. O Turingovu stroji se zmíním v dalším odstavci. Avšak dalším přidáváním zásobníků již výpočetní síla neroste.

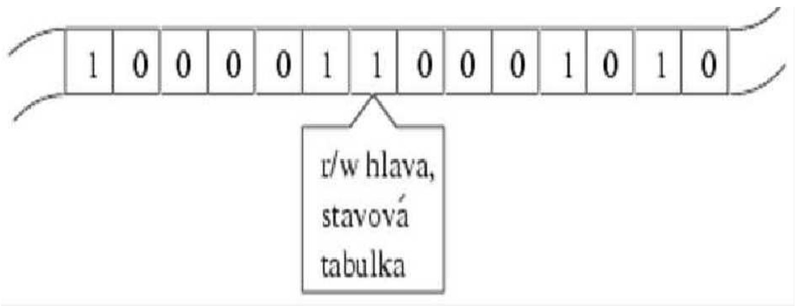
3.3. Turingův stroj

Turingův stroj (Turing machine) je teoretický model počítače, popsaný matematikem Alanem Turingem. Skládá se z procesorové jednotky, tvořené konečným automatem, programu ve tvaru pravidel přechodové funkce a potenciálně nekonečné pásky pro zápis mezivýsledků. Využívá se pro modelování algoritmů v teorii vyčíslitelnosti.

Turingův stroj má několik základních vlastností:

- musel nahradit složitou symboliku matematických kroků. V takovém případě šlo každou konečnou množinu symbolů nahradit pouze dvěma symboly (jako je 0 a 1) a prázdnou mezerou, která by oba symboly oddělovala,
- Turingův stroj má k zápisu nekonečnou pásku skládající se z buněk, do/ze kterých se symbol zapisuje/čte,
- nad touto páskou je možné provádět za pomocí čtecí hlavy operace čtení, zápisu a posunu pásky,
- protože je možné symboly číst, zapisovat nebo se posunovat po pásce, je pro Turingův stroj důležitý vnitřní stav, ve kterém operaci čtení provádíme (čtený symbol a stav tak určují další akci a přechod do dalšího stavu).

Protože se chování tohoto stroje vyvíjí podle tabulky přechodů, můžeme říci, že každý následující stav lze jednoznačně určit na základě čteného symbolu a aktuálního stavu. Jeho chování je proto *deterministické*.



Obrázek 2: *Deterministický Turingův stroj*

Výpočet začíná tak, že jsou na pásce uložena počáteční data a vlastní kód programu. Hlava je pak uvedena do stavu, který odpovídá načtení kódu programu a stroj tak započne výpočet, přechází mezi stavy a po skončení většinou zapíše výsledek. Je zřejmé, že takto se chovající model by se dal velmi dobře přirovnat k funkci dnešních počítačů. Přestože moderní technologie nevídaně od 30.let pokročily, Turingův model můžeme k popisu chování počítačů použít bez úprav i dnes [3].

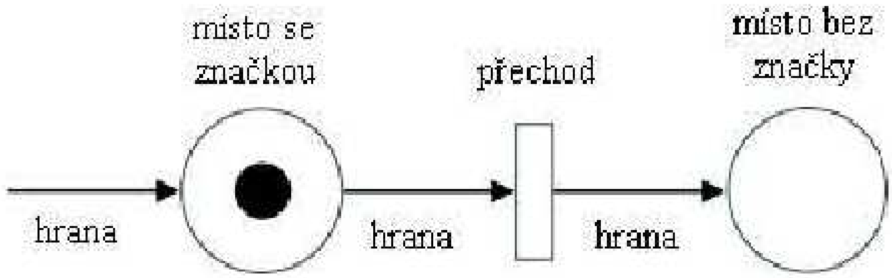
Turingův stroj je v mnoha směrech podobný konečnému automatu, má konečný počet stavů, ve kterých se může nacházet, postupně přijímá jednotlivé vstupy, které dostává ze svého okolí, a reaguje na ně přechodem do nového stavu. Na rozdíl od konečného automatu je ale vybaven navíc ještě i potenciálně nekonečně dlouhou páskou, na kterou si může zapisovat znaky z určité pevně dané abecedy. V každém okamžiku je však na této pásce zapsán jen konečný počet symbolů. Díky nekonečnosti pásky, kterou si Turingův stroj může podle potřeb posouvat, tak má k dispozici nekonečně velkou paměť (obdobně jako zásobníkový automat). Právě díky tomu, a na rozdíl od konečného automatu, je pak schopen namodelovat jakýkoli výpočet, kterého je v principu schopen kterýkoli počítač.

3.4. Petriho sítě

Petriho sítěmi (Petri nets) je v označována široká třída diskrétních matematických modelů (strojů), které umožňují opisovat specifickými prostředky řídící toky a informační závislosti uvnitř modelovaných systémů. Jejich historie je datována od roku 1962, kdy německý matematik C. A. Petri zavedl ve své doktorské disertační práci "Kommunikation mit Automaten" nové koncepty popisu vzájemné závislosti mezi podmínkami a událostmi modelovaného systému. Petriho sítě vznikly za účelem rozšíření modelovacích možností konečných automatů.

Neoznačená Petriho síť je orientovaný ohodnocený bipartitní graf. Petriho síť se tedy skládá z uzlů, které jsou navzájem propojeny hranami. Označená Petriho síť vznikne umístěním značek (tokenů) do míst neoznačené Petriho sítě [4].

- **místo** (*place*) - může obsahovat nezáporný celý počet značek,
- **přechod** (*ransition*) - v okamžiku aktivace (přeskoku) přechodu jsou odebrány značky ze vstupních míst a přidány značky do výstupních míst přechodu,
- **hrana** (*arc*) - propojují místa a přechody.



Obrázek 3: *Prvky Petřino sítě*

Umístění značek v místech Petriho sítě před první aktivací (přeskočením) některého přechodu se nazývá počáteční značení a popisuje počáteční stav systému. Vývoj systému je reprezentován přesunem značek v síti na základě aktivace přechodů. Každé nové značení reprezentuje nový stav systému.

4. VÝSLEDKY A DISKUSE

4.1. Využití konečného automatu pro modelování zpětné vazby

Konečný automat (resp. sekvenční stroj), je dosti silným prostředkem. Postačuje například pro formální popis většiny přenosových protokolů používaných v počítačových sítích. Je ovšem jen jedním z mnoha nástrojů, které si vytvořily teoreticky orientované vědy o počítačích, a není zdaleka prostředkem nejsilnějším. Existují totiž i takové "činnosti" (výpočty, algoritmy), které pomocí konečného automatu namodelovat nelze. Na vině je právě omezení dané konečným počtem stavů automatu, který je díky tomu schopen si pamatovat například jen konečný počet mezivýsledků (resp. konečný počet kroků ze své "historie"). Konečný automat například nelze použít pro namodelování tak banálního výpočtu, jakým je násobení dvou libovolně velkých čísel, neboť při určité velikosti obou činitelů by si již nedokázal zapamatovat potřebné mezivýsledky v tom počtu stavů, které má k dispozici. Obecně lze říci, že každou jednotlivou úlohu, kterou lze řešit pro daná jedna data algoritmicky, tj. počítači, lze řešit i konečným automatem. Neplatí to však o algoritmu jako takovém, který by měl být hromadný, tj. pracovat pro celou, potenciálně nekonečnou množinu daných úloh. Jinými slovy, každý konkrétní algoritmický výpočet prováděný počítačem lze modelovat konečným automatem, avšak neexistuje konečný automat, kterým by bylo možné modelovat každý algoritmický (na nějakém počítači proveditelný) výpočet.

Jedna z vlastností konečného automatu, které jsem nastínila v předešlém odstavci, tj. že konečný automat je schopen zapamatovat si jen konečný počet stavů, omezují jeho využití při modelování výukového systému. Je-li jádro výukového systému tvořené pomocí konečného automatu, nedokáže si systém zapamatovat přirozené číslo a je schopen rozlišovat pouze konečný počet (N) stavů. Pokud by tedy student pro úspěšný průchod výukovým systémem potřeboval více jak těchto N stavů, konečný automat by nebyl schopen všechny tyto stavy uchovat. Při užití konečného automatu pro modelování procesu výuky musí být tedy brána v potaz tato omezení.

Přesný matematický důkaz vyplývá z Nerodovy věty, která se nejčastěji používá v důkazech, že nějaký jazyk není rozpoznatelný konečným automatem. Tedy jazyk je rozpoznatelný konečným automatem tehdy a jen tehdy, jestliže existuje ekvivalence na množině slov, která je invariantní vůči rozšíření zprava a jazyk, který se má rozpoznat je sjednocením z konečně mnoha tříd podle této ekvivalence. Přesná formulace Nerodovy věty je uvedena v [1] a [2]. Pokud bychom tedy chtěli použít konečný automat pro modelování výukového systému (kurzu), bylo by potřeba nejprve stanovit apriorní omezení na počty průchodů kurzem a pokud by student "vyčerpal" tento počet průchodů, kurz by byl ukončen.

4.2. Využití zásobníkového automatu pro modelování zpětné vazby

Zásobníkový automat by v elektronickém výukovém kurzu byl vhodný pro navigaci studenta při průchodu kurzem (neboť je obecně nedeterministický) i pro poskytování zpráv lektorovi (pomocí zásobníku). Patrně ale neposkytuje studentovi při průchodu kurzem vždy potřebnou volnost, neboť možnosti přechodu z daného stavu nezáleží na historii (na kontextu).

4.3. Využití Turingova stroje pro modelování zpětné vazby

Pomocí Turingova stroje lze implementovat každý algoritmus. Proto by bylo možné pomocí Turingova stroje implementovat také algoritmus průchodu studenta elektronickým výukovým kurzem. Pro tento účel by však využití Turingova stroje bylo příliš složité, je proto vhodnější použít jiné, jednodušší modely, například Petriho sítě.

4.4. Využití Petriho sítí pro modelování zpětné vazby

Síťový výukový systém

Dle myšlenek obsažených v [5] a [6] lze výukový systém modelovat pomocí barevné (ohodnocené) Petriho sítě. Síťový výukový systém lze definovat jako množinu všech podsítí, z nichž každá tvoří určitý vyšší logický celek (modul, kapitolu, výukovou lekci, apod.) a každá může být připojena i k další podsítí, která na prezentované učivo navazuje.

Výukový systém se síťovou strukturou vazeb mezi lekcemi umožňuje implementovat vztahy, které jsou součástí prezentovaného učiva. Jednou ze základních podmínek, která musí být při implementaci splněna, je správná návaznost vyučované látky, neboť není vhodné vysvětlovat určitý odvozený pojem, pokud student nezná význam pojmu základního. Dalším principem je možnost aktivního zapojení studenta, u něhož je zapotřebí podle povahy učiva ověřovat teoretické znalosti nebo mu uložit trénování a zmechanizování postupů.

Obyčejná Petriho síť může být použita například pro obecné modelování pojmové sítě daného předmětu. Jeli Petriho síť použita jako řídicí mechanismus výukového systému, umožňuje implementovat nejen zmíněné principy, ale i například variantní cesty výkladu nebo testování, což zmírňuje stereotypní projevy stroje a zvyšuje vypovídací schopnost testů.

Výukový systém navržený za pomoci barevné Petriho sítě lze vystavět jako prázdný výukový systém, jehož naplněním teprve učitel určí charakter vyučovaného nebo zkoušeného předmětu.

Interpretace sítě

Před definováním modelu výukového systému je nezbytné interpretovat všechny prvky barevné (ohodnocené) Petriho sítě vzhledem k vazbám na výukový systém:

- **Místa** - představují pozice výukového systému, které lze přesně charakterizovat určitou množinou zvládnutých pojmů (vymezení pojmů, které má daná výuková množina vysvětlit a protestovat vytváří přesně stanovené místo v síti).
- **Přechody** - představují výukové procedury, každý přechod je jednoznačně charakterizován vysvětlovaným pojmem (nebo množinou pojmů, která je v rámci systému považována za nedělitelný celek a v jiných souvislostech se vždy v jednom celku vyskytuje).

Pro to, aby výukový proces mohl být zahájen z libovolného místa, začíná každý přechod testem porozumění a schopností užít bezprostředně předcházejí pojmy. Po jeho úspěšném průchodu následuje vlastní prezentace. Přechod bez předchůdců (každý jeho uzel náleží do množiny "počátečních uzlů") neobsahuje test, ale pouze odkazy na množinu pojmů, které se pokládají vzhledem k danému systému za všeobecně známé (např. cílové pojmy jiné podsítě). Výstup systému pak tvoří přechody, z nichž vycházejí pouze "koncové uzly". Tyto uzly obsahují pouze test, který ověřuje dosažení cíle podsítě. Při vynechání prezentace pojmů v určité množině přechodů systém pouze diagnostikuje vědomosti studenta.

Test u přechodu ověřuje znalost všech bezprostředně předchozích pojmů. Jestliže je v testu znalost určitého pojmu z množiny bezprostředních předchůdců úspěšně ověřena, příslušný uzel dostane značku. V opačném případě je proces výuky/examinace přesunut do uzlu, který odpovídá nezvládnutému pojmu. Tímto postupem lze při čisté examinaci z jakéhokoliv počátečního uzlu dospět do stavu, kdy bude označena množina právě těch uzlů, jejichž pojmy student úspěšně zvládnul.

- **Značky** - označení stavu znázorňuje úspěšné dosažení daného stavu, cílem průchodu systémem (výukového nebo examinačního tahu) je označení celé množiny stavů, kterou stanoví učitel. Podle charakteru množiny skutečně označených stavů lze určit obsah zvládnutého učiva.

Podsíť, směřující z množiny určitých výchozích stavů k určitému koncovému stavu, lze nazvat "výukovým tahem". Výukový tah charakterizuje samostatnou část učiva, kterou chce pedagog prezentovat (např. výuková lekce, vyučovací blok, jedna kapitola). Dále lze definovat "examinační tah", tj. takový tah, jehož všechny přechody mají blokováno prezentování pojmů. Doplňkem k tahu předchozímu je "prezentační tah", který lze definovat jako množinu přechodů, které mají blokováno testování.

5. ZÁVĚR

K modelování zpětné vazby elektronického výukového systému lze využít všechny výše jmenované matematické modely. Využití prvních tří z nich však skýtá četná omezení. Při použití konečného automatu musíme definovat jen omezený počet stavů (reprezentující průchody studenta kurzem) a po vyčerpání tohoto počtu stavů kurz ukončit. Zásobníkový automat neposkytuje studentovi při průchodu kurzem potřebnou volnost. Pomocí Turingova stroje lze modelovat jakýkoliv algoritmus, tedy i průchod studenta a generování zpětné vazby. Pro tento účel by však využití Turingova stroje bylo příliš složité, je proto vhodnější použít jiné, jednodušší modely, například Petriho sítě. Nejvhodnějším typem Petriho sítí pro modelování výukového systému jsou ohodnocené (barevné) Petriho sítě. Výukový systém navržený za pomoci barevné Petriho sítě lze vystavět jako prázdný výukový systém, jehož naplněním teprve učitel určí charakter vyučovaného nebo zkoušeného předmětu.

Literatura

[1] KOCUR Pavel, "Úvod do teorie konečných automatů a formálních jazyků", *Západočeská univerzita, Plzeň*, 104 s., ISBN 80-7082-813-7, 2001

[2] HOPCROFT John E., ULLMAN Jeffrey D., "Formálne jazyky a automaty", *Alfa, Bratislava*, 342 s., ISBN 63-096-78, 1978,

[3] KUPČA Vojtěch, "Teorie a perspektiva kvantových počítačů - Klasický Turingův stroj", *[online] - http://cml.fsv.cvut.cz/ kupca/qc/node13.html*,2001,

[4] ČEŠKA Milan, "Petriho sítě", *Akademické nakladatelství CERM, Brno*, 94 s., ISBN 80-85867-35-4, 1994.

[5] ŠORM Milan, "Systém pro podporu distančního vzdělávání", *Masarykova univerzita, Brno*, 88 s., Rigorózní práce. Kapitola 8, *Modelování průchodu studenta studiem*, s. 64 - 72, 2003

[6] ČERNÁ Hana, "Návrh a implementace sít'ového výukového systému", *MZLU, Brno*, 83 s., - Diplomová práce. Kapitola 2, *Informační systémy ve výuce*, s. 8 - 12, Kapitola 4, *Implementace sít'ového výukového systému*, s. 16 - 26, 1996.

Ústav Informatiky AV ČR
DOKTORANDSKÝ DEN '05

Vydal
MATFYZPRESS
vydavatelství
Matematicko-fyzikální fakulty
University Karlovy
Sokolovská 83, 186 75 Praha 8
jako svou *– not yet –* . publikaci

Obálku navrhl František Hakl

Z předloh připravených v systému LaTeX
vytisklo Reprostředisko MFF UK
Sokolovská 83, 186 75 Praha 8

Vydání první

Praha 2005

ISBN *– not yet –*