O výpočetních aspektech vyhledávání modelů

Tomáš Havránek
Středisko výpočetní techniky ČSAV
Pod vodárenskou věží 2
182 07 Praha 8

Účelem přednášky je upozornit na některé aspekty implementace metod analýzy dat, které mohou v blízké budoucnosti hrát důležitou roli zejména proto, že by i u nás mohla být dostupná výpočetní technika, o které jsme si zatím mohli pouze číst.

Jak bylo zvykem při obvyklých metodách programování, byly zpravidla v minulosti procesy probíhající při práci programů realizujících procedury analýzy dat chápány zcela sekvenčně. Příkladem, který jsme v minulosti analyzovali velmi podrobně, jsou procedury metody GUHA, např. IMPL nebo ASSOC (Hájek a kol., 1983, Hájek, 1984).

I při skriktně sekvenčním chápání je možné rozeznávat v práci procedury GUHA i jiných procedur analýzy dat tři typy procesů:

1) **symbolické procesy** (realizující např. generování modelů - sentencí)

2) **datové procesy** (ověřování modelů v datech)

3) **procesy vstupu a výstupu** .

Při podrobnějším pohledu (srovnej i jiné procedury tohoto typu, např. Edwards a Havránek, 1985, 1987) je vhodné rozeznávat

2a) **numerické procesy** (např. výpočty různých statistik z frekvencí)

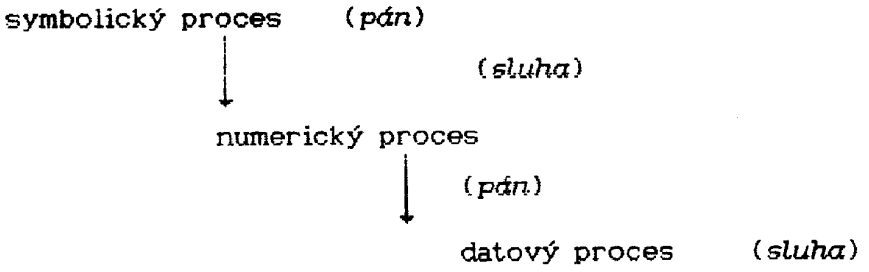
a

2b) **(vlastní) datové procesy** (např. výpočty frekvencí).

Pro numerické procesy při ověřování modelů v datech se používá již kondenzované informace - nezasahují přímo do datové matice. V případě implikační procedury GUHA-IMPL s jednoduchým kvantifikátorem $\Rightarrow_{p,s}$ jsou ovšem tyto procesy nevýrazné (velmi jednoduché). Vlastní datové procesy obstáravají pořizování kondenzované informace z datové matice (např. výpočet postačujících statistik, na kterých je založeno další vyhodnocování).

Všimněme si toho, že stávající pracující verze implikační i asociační procedury jsou realizovány na IBM 370/JSEP 2 i na IBM PC v jazyce FORTRAN, který svým charakterem vyhovuje především numerickým procesům - část datových procesů bylo nutné kvůli efektivitě realizovat v Assembleru. Symbolické procesy jsou ve FOTRANu realizovány obtížně: již jen např. konjunkce jsou reprezentovány jako INTEGER pole obsahující čísla vlastností (veličin).

Uvažované procesy mají zřetelnou hierarchii:

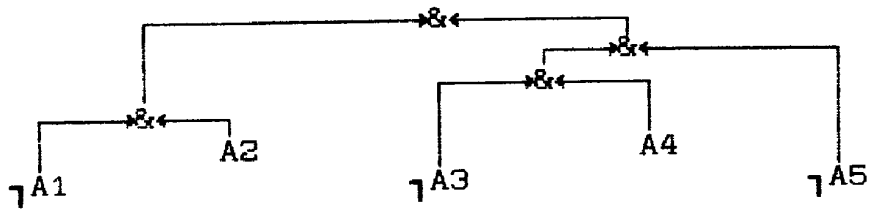


Programové realizace by měly jak trojí charakter těchto procesů, tak jejich hierarchii respektovat a to i za cenu využití tří odlišných programovacích jazyků.

Pro nejvyšší, symbolickou úroveň je možné uvažovat použití vhodně (funkcionálně, objektově) orientovaného jazyka (LISP). Pro nižší úrovně je jistě vhodné použít obvyklé příkazové (imperativní) jazyky. Jako kompromis byl zvolen při nově navržené realizaci implikační GUHA procedury na IBM PC (J. Rauch a kol.) TURBO PASCAL 5.0

Podívejme se nyní na výše uvažovanou strukturu z hlediska paralelismu (viz např. Kober, 1988). Začněme z nejnižší úrovní, t.j. **datovou úrovní**. Při práci s datovou maticí v našem případě jde o SIMD paralelismus (AND a NEG na bitových řetězcích), který může být realizován přídavným procesorovým polem pracujícím na bitových řetězcích. Při úpravě algoritmu je nutné ovšem uvážit cenu komunikace s takovým polem, proto např. je možné opustit koncepci stop (Rauch, 1981) a ohodno covat vždy celé konjukce či disjunkce pomocí posloupností AND, OR a NEG bitových operací realizovaných procesorovým polem (každý procesor v poli může ovšem reali zovat operaci na bitovém řetězci určité pevné délky).

Konjunkci, např, $\neg A_1 \& A_2 \& \neg A_3 \& A_4 \& \neg A_5$, by bylo možné ohodnocovat paralelně (implicitní paralelismus) po uzávorkování. Tedy např. postupovat tak jak je znázorněno v následujícím schématu. Nutné pořadí provádění operaci je pouze částečné:



Jde o přístup s využitím data-flow architektury pro vyhodnocování termů (EM-LISP, Yamagudri, 1984). Tento přístup však nemá rozumnou praktickou cenu v našem kontextu, není-li spojen s paralelním prováděním operací na bitových řetězcích.

Numerickou úroveň nebudeme nyní rozebírat, neboť v realizovaných procedurách (ASSOC a IMPL) nehraje příliš podstatnou roli, s vyjímkou použití kvantifikátorů FISHER, LIMPL a UIMPL (Hájek a kol., 1983). Vstupem pro vyhodnocování těchto kvantifikátorů, stejně jako ostatních, jsou dvě či čtyři frekvence (integer). V daných případech je pak nutné vypočítat určité kombinatorické součty (s daným počtem sčítanců). Při realizaci procedur je výpočet součtů nahrazen výpočtem tabulky (tento výpočet se provede jednou před vlastním během jádra procedury) a vyhodnocování se provádí srovnáním s číslem v tabulce. Obecně vzato byly ovšem dosud úvahy o paralelizaci ve statistických výpočtech zaměřeny právě na numerické procesy (viz např. Scherwish, 1988), zejména na maticové operace (Stewart, 1988, Van de Vorst, 1989, Havránek a Strakoš, 1989); viz dále.

Pokud se týká **symbolické úrovně**, je vhodné uvažovat o paralelismu při jejím styku s numerickými procesy (resp. datovými procesy). Např. při práci podprogramu IMPR, realizujícího "zlepšování", je nutné ověřit (v datech) jistou podmínku týkající se například konjunkcí (kombinovaných vlastností) φ, ψ a daného seznamu literálů $L_1,\ldots,L_n$ (literál je vlastnost či její negace). Jde o typický případ pro použití instrukce typu PCALL (Multilisp, např. Halsted, 1986), t.j. explicitní fork-operace:

$$\text{PCALL } (F(\varphi,\psi,.) \ , \ L_1,\ldots,L_n).$$

Adekvátní by byla multiprocesorová realizace se sdílenou pamětí (data nemodifikujeme!). V každé větvi se zde využívá jedna nebo dvě operace na bitových řetězcích. Podobná situace je při ověřování prostoty (PRIME).

Na vyšší úrovni vlastního GEN procesu (generujícíhé sentence) je možné uvážit, které sentence jsou nezávislé v tom smysly, že pravdivost jedné neovlivňuje nutnost ověřovat pravdivost druhé, či její prostotu.

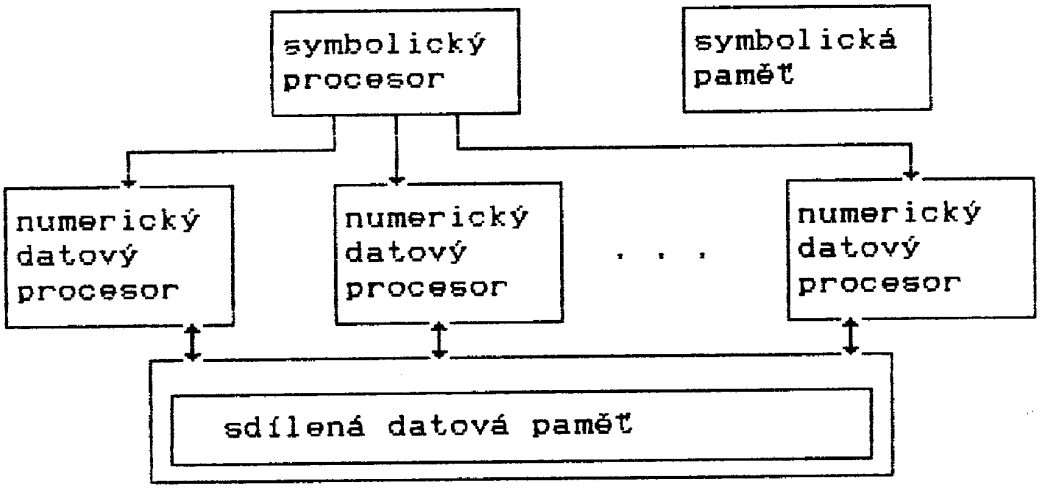
Při poněkud zjednodušeném pohledu to budou sentence s danou délkou antecedentu a sukcedentu; nechť jsou to sentence $\varphi_1 \Rightarrow \psi_1$, $\varphi_1 \Rightarrow \psi_2$, ... , $\varphi_n \Rightarrow \psi_m$. Pak by bylo možné uvažovat explicitní fork-operaci

$$\text{PCALL } (\text{EVAL}; \varphi_1 \Rightarrow \psi_1,\ldots,\varphi_n \Rightarrow \psi_m),$$

ale vzhledem k počtu sentencí by se paralelismus z praktického hlediska rozbíhal do šíře. Např. pro případ, kdy uvažujeme antecendenty délky 3, sukcedenty délky 2, různé antecedentové a sukcedentové vlastnosti v hledaném tvaru, máme při 20 antecedentových a 10 sukcedentových vlastnostech $\binom{20}{3} * \binom{10}{2} = 1140 * 45 = 51300$ sentencí. Takto široký paralelismus by byl na multiprocesorových systémech těžko realizovatelný. Vhodné by ovšem mohlo být použití procesorových polí s elementy odpovídající složitosti a počtem těchto elementů řádově srovnatelným s počtem sentencí (Connection Machine, Loral MPP).

Podotkněme, že fork-paralelismus a paralelní realizace operací na bitových řetězcích jdou do jisté míry "proti sobě". Každému procesoru pole realizujícího fork-paralelismus by muselo být přiřazeno "přídavné" procesorové pole s jedno duchými elementy pro paralelní realizaci bitových operací. Paralelní realizace operací na bitových řetězcích však může být účinná pouze při delších sentencích (díky nutné komunikaci); toto by nenastávalo ve velkém množství případů sentencí ohodnocovaných paralelně "fork" polem.

V zásadě je možné dospět k názoru, že pro realizaci procedur GUHA by byla vhodná hierarchická multiprocesorová architektura s jedním řídícím procesorem (realizujícím symbolický proces), jemu podléhajícími numerickými (datovými) procesory, vybavenými přídavnými poli pro operace na bitových řetězcích a sdílenou datovou pamětí schematicky znázorněná na obr. 1.



Obr. 1

V dalším výkladu uvidíme, nakolik by taková architektura byla vhodná i pro další procedury získavající znalosti z dat. Jako reálný vzor této architektury může sloužit systém Cedar (Kamath et al., 1985) s globálním řídícím procesorem a shluky podřízených procesorů.

Poznamenejme, že symbolická úroveň procedur pro získávání znalostí z dat je v obvyklých systémech statistických programů potlačena. Zajímavé je, že i programy či programové systémy

využívající LISP (Tierney, 1989, Karjalainen, 1989a,b) nevěnují pozornost symbolické úrovni procesů ale spíše se zaměřují na implementaci aritmetických funkcí v LISPu tak, aby bylo možné statistické programy (v numerické a datové urovni) flexibilně vytvářet a modifikovat a přitom využívat pro analýzu dat lispovské stroje. Jde o to tedy zpravidla o to, využívat lispovský styl pro řešení klasických úloh. Náš přístup je tedy odlišný.

Obraťme se nyní od procedur metody GUHA k věcem, které byly na seminářích ROBUST několikrát probírány, tj. k metodám vyhledávání například grafových modelů pro kategoriální data.

Označme $\mathfrak{M}_1$ množinu grafových modelů (mohlo by jít, jak víme i o jinou množinu modelů mající podobné algebraické vlastnosti). Nechť nyní $S = \{m_1, \ldots, m_p\}$, $m_i \in \mathfrak{M}_1$ pro $i = 1, \ldots, p$ je množina grafových modelů a předpokládejme, že tyto modely jsou akceptovány. Množina modelů z $\mathfrak{M}_1$, které můžeme pokládat za slabě akceptované je

$$S^+ = \{m \in \mathfrak{M}_1;\ m_i \leq m \text{ pro některé } m_i \in S\};$$

o modelech z

$${}^cS^+ = \{m \in \mathfrak{M}_1; \text{neplatí } m_i \leq m \text{ pro } i=1,\ldots,p\}$$

nemůžeme říci nic. Přitom ${}^cS^+ = \mathfrak{M}_1 - S^+$.
Položme nyní

$$D_r(S) = \max {}^cS^+$$

(max (A) je množina maximálních prvků množiny A vzhledem k uspořádání $\leq$). $D_r(S)$ nazýváme r-*duál* množiny S. Jsou to nejsložitější modely, které můžeme ještě zamítnout, jsou- li akceptovány modely v S. Všimněme si, že jestliže by byly modely z $D_r(S)$ zamítnuty, pak všechny modely z ${}^cS^+ - D_r(S)$ jsou slabě zamítnuté a každý model z $\mathfrak{M}_1$ je již akceptován či slabě akceptován nebo zamítnut či slabě zamítnut. Množiny S, $S^+ - S$, $D_r(S)$ a ${}^cS^+$ tvoří disjunktní rozklad S.

Podobně můžeme definovat a-*duál* množiny S: nejprve položíme

$$S^- = \{m \in \mathfrak{M}_1;\ m \leq m_i \text{ pro některé } m_i \in S\}$$

(množina slabě zamítnutých modelů, jsou-li modely z S **zamítnuty**) a a-duál je pak

$$D_a(S) = \min {}^cS^-$$

kde ${}^cS^- = \mathfrak{M}_1 - S^- = \{m \in \mathfrak{M}_1; m \leq m_i,\ i=1,\ldots,p\}$.

Platí, že ${}^{c}S^{+} = (D_r(S))^{-}$ a ${}^{c}S^{-} = (D_a(S))^{+}$.

Vytváření $D_r(S)$ i $D_a(S)$ je **asociativní** v následujícím smyslu: $D_r(S_1 \cup S_2) = \max\{s \wedge t;\ s \in D_r(S_1),\ t \in D_r(S)\}$ a $D_a(S_1 \cup S_2) = \min\{s \vee t;\ s \in D_a(S_1),\ t \in D_a(S_2)\}$. Horáková (1989) definuje takto novou operaci: $D_r(S_1 \cup S_2) = D_r(S_1) \hat{\wedge} D_r(S)$ a $D_a(S_1 \cup S_2) = D_a(S_1) \hat{\wedge} D_a(S_2)$.

Popišme nyní formálněji navržený algoritmus (*):

V každém kroku jsou A a R množiny modelů, které byly **akceptovány** resp. **zamítnuty** ($A \cap R = \emptyset$).

Krok 1: Vstupem je nesrovnatelná množina modelů S_0.
Modely z S_0 jsou klasifikovány do A a R (přitom $A \cup R = S_0$, $A \cap R = \emptyset$).

Krok 2: Je-li $A = \emptyset$ jdi na krok 4, je-li $R = \emptyset$ jdi na krok 3, jinak zvol mezi krokem 3 a krokem 4.

Krok 3: Klasifikuj modely z $D_r(A)-R$ jako zamítnuté (R_1) resp. akceptované (A_1).
Je-li $D_r(A) = R_1$, **stop**, jinak polož $A := A \cup A_1$, $R := R \cup R_1$.

Krok 4: Klasifikuj modely z $D_a(R)-A$ jako zamítnuté (R_1) či akceptované (A_1).
Je-li $D_a(R)-A = A_1$, **stop**, jinak $A = A \cup A_1$, $R = R \cup R_1$.

Proceduru jsme popsali jako **symbolickou** manipulaci, bez ohledu na způsob jakým jsou modely klasifikovány jako zamítnuté či akceptované.

Výsledkem procedury jsou množiny A a R takové, že $\mathcal{K}_1 = A^{+} \cup R^{-}$. Z intuitivního hlediska je zřejmé, že by nebylo dobré, aby některý model byl současně (slabě) akceptován či (slabě) zamítnut. Musí být tedy $A^{+} \cap R^{-} = \emptyset$.

Platnost $A^{+} \cap R^{-} = \emptyset$ je zaručena, jestliže pro žádný model $m_1 \in A$ a žádný model $m_2 \in R$ neplatí $m_1 \leq m_2$.

Postačuje, aby tato podmínka byla splněna pro $A \cup R = S_0$ v prvním kroku, kroky 3 a 4 ji již nemohou porušit, např. pro $m \in D_r(A) - R$ je $m \in \max({}^{c}A^{+})$ a tedy nemůže být $m \leq m'$ pro $m' \in R$ ani $m'' \leq m$ pro $m'' \in A$; bez ohledu na klasifikaci m jako přijatého či zamítnutého nemůže být podmínka porušena. Nejjednodušší a interpre tačně racionální je právě zvolit S_0 jako **nesrovnatelné**.

Konstatujme tři důležitá fakta o diskutované proceduře:

(i) Procedura skončí po konečném počtu kroků (neboť A a R monotonně rostou).

(ii) Není potřebné uchovávat A a R celé, stačí minimální a maximální prvky. V prvním kroku je $A := \min(A)$ a $R := \max(R)$. V kroku 3 a 4 pak klademe $A := \min(A \cup A_1)$ a $R := \max(R \cup R_1)$. Takto definované A a R již neobsahuje všechny akceptované či zamítnuté modely, ale stále je $A^+ \cup R^- = \mathcal{M}_1$.

(iii) Díky asociativitě je při hledání duálů možné duály pouze po kroku 3 a 4 upravovat, $D_r(A \cup A_1) = D_r(A) \hat{\wedge} D_r(A_1 - A)$, $D_a(R \cup R_1) = D_a(R) \check{\vee} D_a(R_1 - R)$.

Volba v kroku 2 mezi krokem 3 a krokem 4 závisí na **ceně**, (ve smyslu např. počítačového času), jakou připisujeme klasifikaci modelu. Je-li klasifikování modelů drahé (nezávisle na modelu) je vhodné volit podle velikosti $D_r(A) - R$ resp. $D_a(R) - A$. Obecně závisí výsledné A a R na volbě S_0 a na rozhodnutích v kroku 2.

Zopakujme, že v algoritmu (*) používáme dvou pravidel, které mají vlastně za účel **nahradit numerické/datové výpočty výpočty symbolickými**. Je definován pojem slabě zamítnutých a slabě akceptovaných modelů. Otázkou je, kdy výše zmíněná náhrada je zcela adekvátní.

Podmínkou proto je požadavek, aby rozhodovací pravidlo d, klasifikující modely při daných datech **M** jako akceptované (d(m,**M**) = a) či zamítnuté (d(m,**M**) = r) bylo **koherentní**:

pro žádní data **M** a pro žádný pár modelů m_1 a m_2 takových, že $m_1 \leq m_2$ nesmí nastat d(m,**M**) = a a $d(m_2, \mathbf{M}) = r$.

Je-li rozhodovací pravidlo koherentní, pak:

(A) Model m je slabě zamítnut právě tehdy, může-li být zamítnut; podobně pro akceptování. Není tedy z interpretačního hlediska nutné rozlišovat mezi slabě zamítnutými a zamítnutými modely.

V důsledku toho platí dále:

(B) Výsledek práce algoritmu (*) nezávisí na volbě počáteční nesrovnatelné množiny modelů S_0, ani na rozhodování v kroku 2 mezi krokem 3 a 4.

Výsledek práce algoritmu je tedy při daných datech jednoznačně dán.

Je nutné si uvědomit, že výsledek práce je sice jednoznačně dán, ale počet ověřovaných modelů v datech, kterým můžeme měřit složitost práce algoritmu závisí při daných datech silně na volbě počáteční množiny S_0 i na rozhodnutích v kroce 2.

Zhruba řečeno, záleží na tom, jak je S_0 blízké minimálním akceptovaným modelům (je-li např. $S_0 = A$, práce končí). Proto je vhodné volit počáteční množinu modelů na základě některých předběžných testů (viz Benedetti a Brown, 1978).

Doporučená volba (volit 3 nebo 4) podle srovnání počtu modelů v $D_r(A)$-R resp. $D_a(R)$-A minimalizuje počet ověřovaných modelů v dalším kroku algoritmu, nikoliv však celkový počet ověřovaných modelů.

Naneštěstí v námi zatím uvažovaném případě grafových či HLL modelů, není obvyklé rozhodovací pravidlo

$$d(m,\mathbf{M}) = r, \text{ je-li } G^2(m,\mathbf{M}) \geq \chi^2(m) \quad (**)$$

koherentní. Pravidlo lze změnit na koherentní, nahradíme-li $\chi^2(m)$ závislé na počtu stupňů volnosti (na modelu) pevnou mezí, řekněme K. Protože platí, že $G^2(m_1,\mathbf{M}) \geq G^2(m_2,\mathbf{M})$ pro $m_1 \leq m_2$, pravidlo

$$d'(m,\mathbf{M}) = r, \text{ je-li } G^2(m,\mathbf{M}) \geq K$$

je pak koherentní. Toto pravidlo má velkou nevýhodu, má totiž malou rozlišovací schopnost (sílu), je-li K zvoleno tak, aby pro každý model šlo o (konzervativní) test dobré shody na hladině α (viz Havránek a Soudský, 1989): s velkou pravděpodobností může docházet K nezamítnutí "nesprávného" modelu. Stačí si uvědomit, že pak pro většinu modelů je $x^2(m) \ll K$.

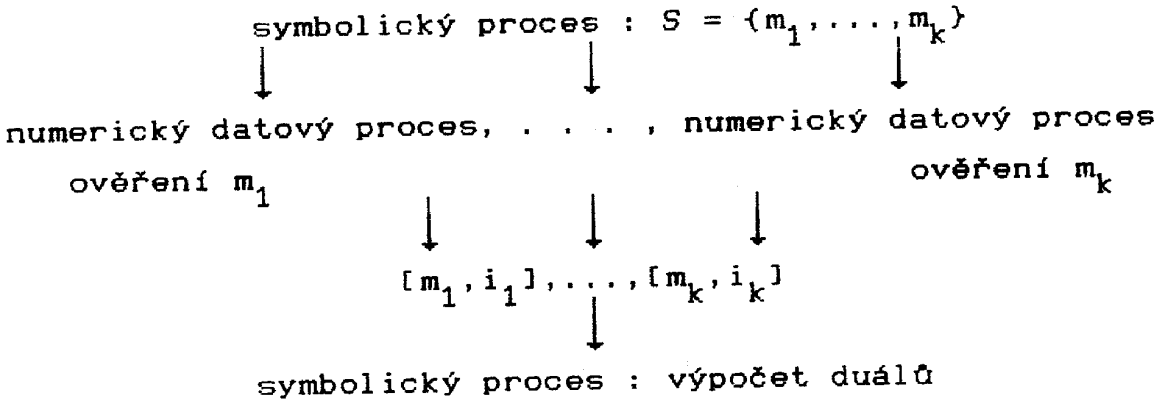
Na druhé straně, na základě zkušenosti je možné soudit, že pravidlo (9) je **kvasikoherentní** (Edwards, Havránek, 1985). To znamená, že porušení koherentnosti není příliš časté. Tato otázka není dosud zdaleka prozkoumána; problém je komplikován tím, že je zde nutné počítat různé pravděpodobnosti zejména pro necentrální χ^2-rozložení. Dílčí výsledky jsou obsaženy v práci (Havránek, Pokorný, 1985).

Uvažujme nyní pro jednoduchost dále **koherentní případ.** Nejjednodušší forma paralelismu může při práci algoritmu (*) spočívat v tom, že v kroce 3 resp. 4 je vždy volána procedura ověření modelu v datech souběžně. Jde o klasickou fork operaci:

$$\text{PCALL (ověření; } m_1,\ldots,m_k),$$

kde $m_1,\ldots,m_k$ jsou modely z S_0, $D_r(A)$-R či $D_a(R)$-A. Každá paralelně probíhající operace není ovšem primitivní; pro daný model m_i je vždy nutné vytvořit ze sdílených (a nemodifikovaných dat) protřebné marginální tabulky a pak aplikovatovat proceduru odhadu parametrů pro daný model a vyhodnotit rozhodovací kriterium (jde tedy o situaci vhodnou pro multiprocesorový

systém se sdílenou pamětí a bez synchronizace). Symbolický proces čeká na "návrat" všech modelů, pak provádí výpočet duálů:



(zde i_j je a nebo r).

Jak již jsme řekli výše, numerické/datové procesy si musí vytvořit z celkové tabulky svoje vlastní kopie marginálních tabulek; pak již mohou pracovat (např. iterovat) zcela asynchronně (srovnej hierarchickou architekturu na obr. 1).

Výpočet a-duálu může být realizován voláním následující schématicky popsané rekursivní procedury (Horáková, 1989a):

procedura a-duál množiny modelů:
(var S : set of models, var h = $|S|$; integer);
begin $D_a(S) := 0$;
$S_1 := (m_1, m_2, \ldots, m_{h/2})$;
$S_2 := (m_{h/2+1}, \ldots, m_k)$;
if $|S_1| > 1$ then a-duál množiny modelů $(S_1, h/2)$
else if $|S_1| = 1$ then a-duál modelu (S_1);
if $|S_2| > 1$ then a-duál množiny modelů $(S_2, h/2)$;
else it $|S_2| = 1$ then a-duál modelů (S_2);
end;

Podobně pro r-duál. Jde tedy o proces typu binárního stromu.

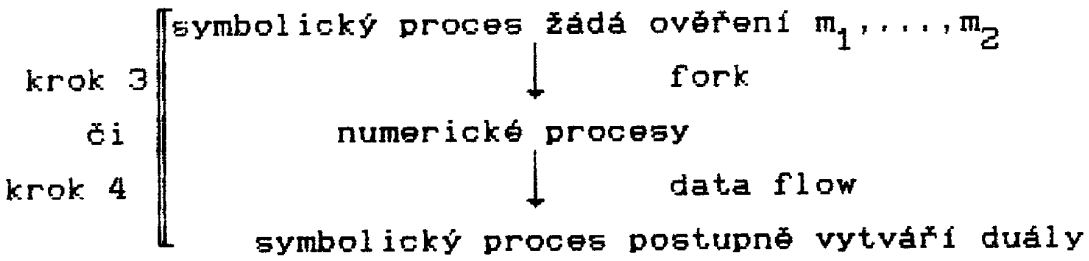
Při důsledné paralelizaci algoritmu (*) je však zřejmě výhodnější jiná cesta: k "návratu" ohodnocených modelů z numerických procesů $p_1(m_1), \ldots, p_k(m_k)$ nedochází v jednom čase, ale postupně; je tedy výhodnější, aby symbolický proces nečekal na návrat všech modelů, ale pracoval vždy po návratu libovolného modelu $[m_j, i_j]$ (data flow) s použitím

(a) výpočtu duálu modelu $D_a(m_j)$ nebo $D_r(m_j)$ podle hodnoty i_j,

(b) aplikace operace $D_a(R) \overset{\vee}{\vee} D_a(m_j)$ resp. $D_r(a) \overset{\wedge}{\wedge} D_r(m_j)$.

Bod (a) a (b) probíhá pouze, je-li ověřeno, že není $m_j \leq m$ pro nějaké $m \in R$, resp. $m_j \geq m$ pro nějaké $m \in A$ (přitom probíhá odstraňování redundatních modelů v R i A).

Dojde-li v průběhu (a), (b) k návratu dalšího modelu, je uložen v zásobníku a zpracován až po ukončení práce symbolického procesu pro předchozí model. Zřejmě je možná situace, kdy pak je vhodné zpracovat více modelů zamítnutých (R_1) či akceptovaných (A_1) najednou. Pak je možné, aby symbolický proces (po ověření nerendundance) aplikovat rekursivní proceduru na $D_a(R_1)$ resp. $D_u(A_1)$ v bodě (a) a pak v (b) použil $D_a(R_1)$ místo $D_a(m_j)$. Situace je tedy:



Vytvoření duálů je tedy řízeno návratem informace o zamítnutí či akceptování modelů (data flow); díky "asociativitě" není nutná synchronizace.

Literatura

Benedetti, J.K., Brown, M.B. (1978): Strategies for the selection of log-linear models. Biometrics 34, 680-686.

Edwards, D. a Havránek, T. (1985): A fast procedure for model search in multidimensional contingency tables, Biometrika 72, 339-351.

Edwards, D., Havránek, T. (1987): A fast model selection procedure for large families of models. J. Amer. Statist. Assoc. 82, 205- 213.

Hájek, P. (1984): The new version of the GUHA procedure ASSOC, COMPSTAT'84, Physica-Verlag, Wien 360-365.

Hájek, P., Havránek, T., Chytil, M. (1983): Metoda GUHA, Academia. Praha.

Halsted, R.M., Jr. (1986): Parallel Symbolic Computing, Computer, No.8, 35-43.

Havránek, T., Pokorný, D. (1985): On the GUHA approach to model in connection to generalized linear models, Generalized linear models, R. Gilchrist, B.

Francis, J. Whittaker (eds), Lecture Notes in Statistic Springer-Verlag, Heidelberg, 82-92..

Havránek, T., Soudský, O. (1989): Model choice in the context of simultaneous inference, in: Y. Dodge(ed.), International Conference in Honor of C. R. Rao, North Holland, Amsterdam, 165-176.

Havránek, T., Strakoš, Z. (1989): On practical experience with parallel processing of linear models, 47th Session of ISI, Bulletin of ISI 53, 105-117.

Horáková, M. (1989a): Dependence structure selection using mixed interaction models, Comp. Statistics Quanterly (zasláno do tisku).

Horáková, M. (1988b): Struktura závislosti dat, kandidátská disertační práce, SVT ČSAV, Praha.

Kamath, C., Sameh, A. H., Yang, G. C, Kuck, D. J. (1985): Structural computations on the CEDAR system, Computers and structures 20, 47-54.

Karjalainen, M. (1989a): A Lisp-based high-level programming enviroment for the TMS320C30, Proc. of IEEE ICASSP-89, Glasgow.

Karjalainen, M. (1989b): Some recent trends in AI software and hardware, D. Zicha (ed.), Význam metod umělé inteligence pro zkoumání biologických zákonitostí, ČSVTS Praha, 1-12.

Kober, R. (1988): Parallelrechner-Architekturen, Springer-Verlag, Heidelberg.

Rauch, J. (1978): Some remarks on computer realisations of GUHA procedures, Int. J. Man-Machine Studies 10, 23-28.

Schervish, M. J. (1988): Applications of parallel computation to statistical inference, J. Amer. Stat. Assoc. 83, 976-983.

Stewart, G. W. (1988): Parallel linear algebra in statistical computations, COMPSTAT'88, Physica-Verlag, Heidelberg, 3-16.

Tierney, L. (1989): LISP-STAT - a statistical enviroment based on the Lisp language, Bulletin of the ISI,

Van de Vorst, J. G. G. (1989): Solving the least squares problem using a parallel linear algebra library, Future Generation Computer Systems 4, 293-297.

Yamaguchi, Y. (1984): EM-3: a lisp based data driven machine, Techn. Report Electronical Laboratory, Ibaunki.