Ř Í D Í C Í J A Z Y K Y P R O G R A M O V Ý C H S Y S T É M Ů P R O A N A L Ý Z U D A T

Josef T v r d í k , Ústav ekologie průmyslové krajiny ČSAV
Miroslav L i š k a , OKR Automatizace řízení, k.ú.o.

1. Úvod

Možnost využití počítačů byla silným impulsem pro rozvoj a aplikace matematicko-statistických metod. Počítače se staly dokonce nezbytným nástrojem pro realizaci moderních matematicko-statistických metod, založených na takovém matematickém aparátu, jehož výpočtová náročnost způsobuje, že jsou manuálně prakticky neproveditelné (např. maticové operace v mnohorozměrných metodách). Počítače ovlivnily i rozvoj a rozšíření používání metod přesahujících rámec klasického pojetí matematicko-statistických metod (jako např. faktorová analýza, shluková analýza, GUHA a pod.). Vznikla tak nová obecnější disciplina - analýza dat (viz např. Tukey, 1977). Analýza dat přinesla pochopitelně další nové problémy metodologické (vymezení oblasti aplikovatelnosti jednotlivých metod, interpretace výsledků a pod.), dále problémy matematické (hledání nejvhodnějších algoritmů, numerických metod, snížení výpočtové náročnosti a pod.) a neméně závažné problémy technické (týkající se zejména programování, organizace a řízení zpracování na počítači atd.). Většina nových technických problémů se objevila v souvislosti s tím, že čas uživatele spojený s výpočty při aplikacích metod analýzy dat se stal zanedbatelným a nejnáročnější činností se stalo zabezpečení vstupu dat do počítače, manipulace s daty na pamětích počítače (data handling) a volba algoritmu výpočtu pro danou úlohu. V tomto příspěvku se zaměříme na některé přístupy k řešení těchto technických problémů.

Vývoj programového vybavení pro analýzu dat začal tvorbou specializovaných programů pro konkrétní úzce vymezené úlohy, pokračoval sestavováním problémově orientovaných knihoven podprogramů se standardizovaným interfacem a dokumentací, až v současnosti dospívá do stadia vývoje a praktického provozování obecných programů nebo ucelených programových systémů. Od takových programových systémů se očekává nejen bohatá nabídka programů pro matematicko-statistické metody, resp. metody analýzy dat, použitelných k řešení různých typů úloh, ale rovněž kvalitní programové zabezpečení vstupu a výstupu dat, umožňujících pružnou a snadnou manipulaci s daty. Snaha o obecnost a univerzálnost takových programů vede k rozšíření vstupních dat o údaje potřebné k popisu datových struktur pro určitý zvolený běh programu. Zajištění vstupu těchto řídících dat (tzv. parametrů) se v současné době provádí většinou pomocí řídících jazyků.

Dnes existuje řada programových systémů pro analýzu dat, které mají zmíněné vlastnosti. Jsou to např. SAS (rozsáhlé možnosti využití grafických výstupů), GENSTAT, MINITAB (původně vyvinut pro výuku metod analýzy dat), SURVO 76 (pro minipočítače WANG). U nás jsou z této kategorie programových systémů známější a na některých počítačích implementovány systémy BMDP (Biomedical Computer Programs - Dixon 1977) nebo SPSS (Statistical Package for Social Science - Nie et al. 1975). Z u nás vytvořených systémů se k této kategorii blíží nová verze systému programů metody GUHA (Havránek 1981 a, b).

Řídící jazyky uvedených programových systémů představují poměrně dosti rozsáhlou a ověřenou ukázku parametrických programů a způsobů zadávání parametrů. Přístupy k řešení užité v těchto systémech mohou být užitečné i mimo oblast tvorby programů pro analýzu dat, případně mohou posloužit aspoň jako inspirace. Z těchto důvodů se jimi budeme v tomto příspěvku zabývat z obecnějších hledisek. Omezíme se na programové systémy pracující v dávkovém režimu. Interaktivní

systémy umožňují totiž kvalitativně jiná a mnohdy jednodušší řešení (komunikace mezi uživatelem a programem prostřednictvím nabízených možností dalšího běhu programu, dialog s odpověďmi Ano - Ne, možnost volání podrobnějších vysvětlení a pod.). Problémy interaktivního řešení programových systémů pro analýzu dat se zabývá např. Tagg (1981), jako příklad uvádí systém SCSS, nebo Miller a Thomas (1977).

2. Parametrizace programů pro analýzu dat

Pokusíme se odpovědět na otázku, kdy je vhodné vytvářet parametrické programy.

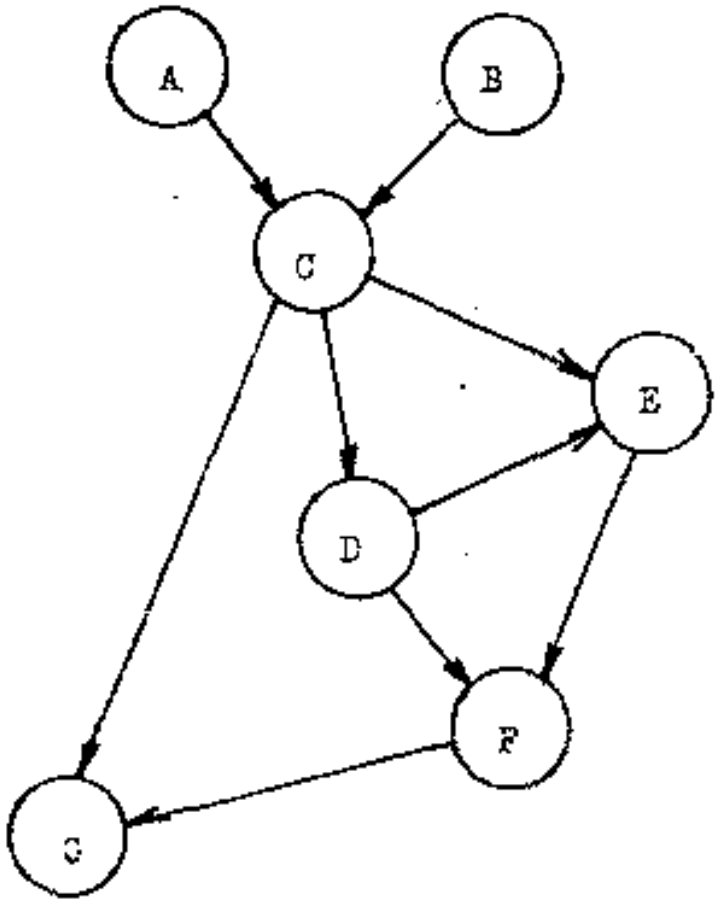
Podle Jiřička (1975) tehdy, když byla pro danou úlohu napsána řada programů speciálních, takže jejich programování se stává jednotvárnou rutinou. V tomto momentu je vhodné přistoupit k obecnějšímu popisu datových struktur, které bude program zpracovávat. K definování těchto datových struktur se zavádí další údaje, které jsou součástí vstupních dat. Těmto údajům říkáme parametry.

Domníváme se, že je vhodné vytvářet obecné parametrické programy také v těch případech, kdy nějaká skupina úloh vyžeduje využití programů sestavených z více funkčních modulů a některé moduly se v jednotlivých programech opakují. Takovou situaci lze znázornit orientovaným grafem (uzly znamenají moduly, hrany možné a smysluplné pořadí volání modulů) - obr. 1. Vidíme, že v grafu na obr. 1 existuje více cest mezi vstupními a výstupními uzly. Každá z těchto cest odpovídá nějakému speciálnímu programu pro řešení dílčí úlohy z dané skupiny úloh. Je zřejmé, že výběr cesty v takovém grafu lze provést podle hodnot vstupních parametrů.

Úlohy analýzy dat splňují obě uvedené podmínky pro tvorbu obecných parametrických programů. Existence celé řady obecných programů (či programových systémů) pro analýzu dat

ukazuje, že lze zvolit různá technická řešení parametrizace programů. Z uvedené charakterizace zavádění parametrů pro obecné programy analýzy dat vyplývá, že parametry jsou dvojího druhu:

- popisující zpracovávané datové modely (budeme jim říkat parametry I. druhu),
- sestavující speciální programový běh (výběr cesty v grafu), těxto budeme říkat parametry II. druhu



Obr. 1 Příklad struktury programu a možného výběru funkcí

A - vstup dat v pevném formátu
B - vstup ze standardního souboru
C - výpočet základních statistických charakteristik pro veličiny
D - třídění dat podle hodnot zadaných veličin
E - tisk vstupních dat
F - uložení na standardní soubor
G - tisk výsledků

3. Řídící jazyky prgramů pro analýzu dat

Účelem a úkolem analýzy dat je zpracování nějakých daných datových struktur získaných měřením nebo pozorováním reality, která je předmětem našeho výzkumného zájmu. Datová struktura bývá nazývána datový model a musí být vždy součástí vstupních dat. Nejčastější formou datového modelu je matice s rozměry - objekty x veličiny.

Způsob, jakým má být datový model zpracován, může být zcela popsán v některém z programovacích jazyků, např. ve Fortranu. Tento jazyk je v tomto případě řídícím jazykem. Do programu nevstupují žádné parametry. Program je speciální ("ušit" na určitou úlohu) a nelze očekávat opakované využití.

Krokem k obecnějším programům je parametrizace programu vzhledem k datovým modelům. Přitom se využívá možností poskytovaných programovacím jazykem, např. dynamická deklarace polí a podpory problémově orientované knihovny podprogramů. Příkladem takového přístupu je užití knihovny SSP (Scientific Subroutine Package) na počítačích IBM a JSEP, nebo podprogramů z knihovny Nag. Součástí vstupních dat jsou parametry I. druhu. Syntax "jazyka", kterým jsou parametry programu sdělovány, je určována možnostmi užitého programovacího jazyka, správný zápis a definování pojmů tohoto "jazyka"

(sémantická stránka) je zcela na uživateli. Parametry druhého druhu do programu nevstupují, výběr funkcí programu zadává uživatel na úrovni zdrojového zápisu programu.

Pro oba uvedené přístupy je společné, že sestavení programu pro řešení dané úlohy je pracné a vyžaduje detailní znalosti programovacího jazyka. Druhým postupem vznikají obecnější programy s větší nadějí na opakované využití.

Dalším vývojovým krokem v parametrizaci programů pro analýzu dat je vytvoření proceduralního řídícího jazyka. Tento přístup se velmi podobá předchozímu. Součástí vstupních dat jsou opět parametry I. druhu. Odlišnost spočívá v tom, že řídící jazyk (analogie pseudokódu) má značně volnější syntaktická pravidla než programovací jazyk a umožňuje rychlejší zápis i uživateli neprogramátorovi. Příkladem takového programového systému pro analýzu dat s procedurálním řídícím jazykem je GENSTAT.

Nejvyšším dosavadním vývojovým stupněm na cestě za uživatelovým pohodlím (při práci v dávkovém režimu zpracování) jsou programové systémy pro analýzu dat vybavené neproceduralním řídícím jazykem. Součástí vstupních dat jsou i parametry II. druhu, takže uživatel může snadno vyjádřit své požadavky na zpracování dané úlohy zadáním požadavků na funkce. Nepotřebuje přitom procedurální uvažování a znalost programovacího jazyka. Syntaktická pravidla těchto jazyků jsou volná, jsou vytvářena zpravidla na dvou úrovních - odstavce a věty. Zvládnutí sémantické stránky je ulehčeno vhodnou mnemotechnikou ve volbě slov jazyka a zavedením default hodnot. Takovým řídícím jazykem jsou vybaveny u nás známé systémy SPSS a BMDP.

Vytvoření řídícího jazyka zohledňujícího potřeby uživatele je však jen jednou z podmínek snadného, efektivního a racionálního využívání programových systémů pro analýzu dat. Pro úspěšné rozšíření programového systému je důležitá celá řada dalších vlastností.

Jednou z nejdůležitějších je dokonalost uživatelské dokumentace a její diferencovanost z hlediska potřeb různých uživatelů (např. stručný manuál pro rutinovaného uživatele, učebnice pro začátečníka, podrobný manuál pro občasného uživatele). Další důležitou vlastností je rozsah a kvalita nabízených funkcí programů. Právě při vytváření programových systémů pro analýzu dat se objevil požadavek na funkce umožnující různé manipulace s daty v paměti počítače. Výběr a řízení těchto funkcí je zapotřebí realizovat parametricky. Jedná se zejména o redukování datového modelu vypouštěním objektů a veličin, vytváření nového datového modelu skládáním více datových modelů primárních, transformace veličin, práce s daty na vnějších pamětech a pod. Kvalitní programové zabezpečení těchto funkcí je z uživatelského hlediska stejně důležité jako rozsah a množství a kvalita programů pro vlastní metody analýzy dat. Další důležitou vlastností programových systémů pro analýzu dat je přehlednost tiskových výstupů, možnost volby různé úrovně podrobnosti výsledků, využití grafických výstupů, datová kompatibilita s jinými systémy a otevřenost programového systému vůči dalšímu zlepšování a poskytování možností pro uživatelská rozšíření. Pro využití většiny současných programových systémů je rovněž důležitou a mnohdy limitující i cenová dostupnost, ale tyto otázky přesahují rámec příspěvku.

Pro další vývoj programových systémů analýzy dat pracujících v dávkovém režimu je důležitá i jedna praktická zkušenost, na kterou chceme zvláště upozornit. Ukazuje se, že většina úloh analýzy dat nevyžaduje rychlou odezvu výpočetního systému na uživatelovy požadavky. Formulace nových požadavků na následný výpočet při řešení nějaké úlohy je zpravidla možná

až po důkladném rozboru výsledků (nejlépe tištěných) z výpočtu předchozího. Interaktivní systémy mohou dokonce někdy tuto důkladnost a komplexní posouzení výsledků narušovat a omezovat. Proto zřejmě dávkové systémy analýzy dat budou i v budoucnosti široce užívány a snaha o zkrácení odezvy bude orientována jen na ty části úlohy, kde je rychlá odezva opravdu žádoucí, např. při syntaktické kontrole příkazů řídícího jazyka.

4. Některé možnosti implementace řídících jazyků

S rostoucí obecností programů pro analýzu dat je spojen i rozvoj metod vstupu parametrů. Patrně nejstarší způsob je zadávání pozičních parametrů v pevném formátu. Vychází z omezení daných Fortranem IV. Je užit např. v programech Cooleyho a Lohnese (1971).

Krokem k neprocedurálním řídícím jazykům s volnějšími syntaktickými pravidly je využití NAMELISTu v rozšířeném Fortranu nebo řízení DATA pro streamový vstup v PL/I. Parametry se v tomto případě zadávají pomocí klíčových slov ve volném formátu. Programové řešení vstupu parametrů a definování default hodnot je velmi jednoduché. Tento způsob je užit např. v některých programech metody GUHA (Havránek, 1981).

U rozsáhlejších programových systémů určených pro široký okruh uživatelů se zpravidla s uvedenými možnostmi nevystačí a především uživatelův komfort vyžaduje zavedení řídícího jazyka pro vstup parametrů. Znamená to definovat syntaktická pravidla, dále významovou stránku jazyka a vytvořit jejich interpretátor. V programovém řešení interpretace parametrů je nutné užít prostředků umožňujících operace s textovými údaji. Např. v PL/I se nabízí pro tyto účely funkce pro

operace nad znakovými řetězci SUBSTR, INDEX, TRANSLATE a pod., nebo příkazy pro přenos dat ve vnitřní paměti: GET, resp. PUT STRING s řízením přenosu pomocí DATA a LIST. Tytéž a celou řadu dalších možností poskytuje ASSEMBLER, zejména po zavedení vhodných maker.

Nezbytnou součástí interpretačního programového modulu je zabudování syntaktických a sémantických kontrol zápisu vstupujících příkazů řídícího jazyka.

Řídící jazyk může poskytovat velmi bohaté výrazové prostředky pro vstup parametrů, aniž by se podstatně zvýšila pracnost programového řešení interpretačního modulu. Je možné dovolit používat synonyma pro klíčová slova, např. ALFA i ALPHA, zavedení zkratek (VARIABLE i VAR), vypouštění samohlásek v klíčových slovech a pod.

Při sestavování interpretačního modulu je důležité si uvědomit, že v interpretaci parametrů I. a II. druhu není rozdíl. I parametry druhého druhu jsou převedeny do takové datové struktury a zobrazení v paměti počítače, aby bylo možné podmíněné volání modulů nebo segmentů programu (tento způsob je užit např. v SPSS a BMDP) nebo dynamické volání modulů z load knihoven (např. makro LOAD Assembleru).

5. Příklad parametrického řídícího jazyka BMDP

Vlastnosti konkrétního neprocedurálního jazyka lze dobře ilustrovat příkladem užití programu P1D z programového systému BMDP (obsahuje více jak 30 obecných programů).

Struktura modulů programu P1D odpovídá přibližně grafu na obr. 1. V příkladu na obr. 2 je zvolena cesta A C D E F G. Příkazy začínající // jsou příkazy JCL, většinu potřebuje

nově vytvářený standardní soubor. Vstupní parametry programu jsou členěny do odstavců uváděných lomítkem a názvem odstavce. V každém odstavci jsou uvedeny věty ukončené tečkou. Části klíčových slov napsaných malými písmeny není nutno uvádět. Kromě toho je možné vypustit samohlásky v klíčovém slovu (až na případ, kdy klíčové slovo začíná samohláskou). Lze užívat množetví synonym. Např.: IS, ARE a = jsou synonyma.

```
//    EXEC    BIMED,PROC=BMDP1D
//FT09F001    DD    UNIT=SYSDA,DSN=jm.souboru,VOL=xxxx,
//                  DISP=(,KEEP),SPACE=(CYL,(1,5),RLSE)
//SYSIN       DD    *
/PROBlem      TITLE = 'UKAZKA'.
/INPut        FORMAT IS '(A4,6X,5F5.0)'. SORT = 'VAHA-1'.
/VARiables    NAMES ARE  ID, VYSKA, 'VAHA-1' , CHOLEST,
              TLAK, ISCH.
              LABEL = 1. .
              MINimum = (2) 140, (3) 30, (5) 80.
              MAXimum = (2) 230, (3) 200, (5) 250.
/CATegory     CODE (6) = 0, 1.
              NAME (6) = NEMA, MA.
/PRINT        DATA.
/SAVE         UNIT = 9. NEW.
/END
.
.
data na DŠ
.
.
/*
```

Obr. 2: Příklad zadání programu BMDP1D.

6. Závěr

Další vývoj programových systémů pro analýzu dat lze očekávat především ve dvou oblastech. Jednak se bude nadále rozšiřovat využívání obecných programových systémů analýzy dat, zvláště ve vazbě na databankové systémy. Vývoj řídících jazyků bude směřovat k jednotnému a z pohledu uživatele jednoduššímu tvaru, respektujícímu jak potřeby řízení databáze, tak programových systémů analýzy dat. Druhou oblastí bude zkvalitňování programové podpory pro rozhodování uživatele ve fázi volby parametrů při řešení konkrétní úlohy analýzy dat. Zde se jeví reálná možnost využití některých výsledků výzkumu umělé inteligence, především v konstrukci "chytrých" programů, které budou uživateli nabízet doporučení na základě automatizované analýzy zkušeností předchozích uživatelů (miniexpertní systémy - Hájek 1982). Objevil se i výzkumný projekt automatizace řešení úloh analýzy dat (GUHA 80 - Hájek, Havránek 1981).

Vedle těchto perspektiv považujeme za nutné připomenout i skutečnosti méně optimistické. Především realizace naznačených směrů vývoje programových prostředků analýzy dat vyžaduje značné množství tvůrčí i rutinní práce nejen v oblasti programování. Navíc prostředky analýzy dat dovolují pouze nahlédnout do daných struktur empirických údajů. I když bude k dispozici technicky dokonalý aparát, umožňující hluboký a mnohostranný pohled do dat, nelze pouze prostředky samotné analýzy dat rozhodnout o tom, zda tvrzení platná v určité datové struktuře mají obecnou platnost (podrobněji viz např. Hájek a Vorlíčková, 1977).

Písemnictví:

Barrit M., Wishart D.,eds, (1980). COMPSTAT 80, Physica Verlag, Wien

Cooley W.W., Lohnes P.R., (1971). Multivariate Data Analysis, Wiley a Sons, Inc

Dixon J.W., ed., (1977). Biomedical Computer Programs, 2.vyd. Univ. of California Press, Los Angeles

Hájek J., Vorlíčková D., (1977). Matematická statistika (skriptum MFF UK Praha), SPN Praha

Hájek P., (1982). Vývoj miniexpertních systémů pro metodu GUHA, přednáška, seminář metody GUHA, Alšovice

Hájek P., Havránek T., (1981). GUHA 80 - An application of AI to data analysis, sdělení MSBU ČSAV č. 25

Havránek T., (1980). Informace z konference COMPSTAT 80, seminář metody GUHA, Alšovice 1980

Havránek T., (1981a). Automatické formování hypotéz metodou GUHA - teorie a aplikace, Pokroky matematiky, fyziky a astronomie 3, 136 - 151

Havránek T., (1981b). Present state of GUHA software, Int. J. Man-Machine Studies 14, 253-264

IBM System/360. Scientific Subroutine Package 5. vyd., 1970

Jiříček P., (1975). Generátory, přednosti a nevýhody této techniky Sborník "Programování počítačů III. gen." Dům techniky ČVTS Ostrava

Miller L.A., Thomas J.C., jr., (1977). Behavioral Issues in the Use of Interactive Systems, Int. J. Men-Machine Studies, 9, 509-536

Nie N., Hull C., Jenkins J., Steinbrenner K., Bent D., (1975). Statistical Package for Social Science, 2.vyd., McGraw Hill, New York

Pokarný D., (1980). Přehled systémů pro analýzu dat, přednáška, seminář metody GUHA, Alšovice 1980

Tagg S.K., (1981). The user interface of the data analysis package: some lines of development, Int. J. Man-Machine Studies 14, 297-316

Tukey J.W., (1977). Exploratory data analysis, Addison-Wesley, Reading