A N A L Ý Z A T O K U D Á T

Ing. Ivan Schnapp, Výpočtové laboratórium SIPK

Úvod

Táto metóda na automatické zostavovanie programov bola prvý raz prezentovaná na Programování '83 /1/. V príspevku sú popísané odvtedy dosiahnuté výsledky. Pretože tu chceme podať pokiaľ možno úplný popis metódy, príspevok sa v niektorých častiach prekrýva s /1/. Na ilustráciu budeme používať príklady z /1/.

Metóda teraz dovoľuje automaticky zostaviť nielen časti programu, ale celý program. Okrem toho umožňuje automaticky deklarovať /tj. navrhnúť štruktúru a vytvoriť popis/ vstupných, výstupných a maticných súborov.

Obmedzenia

Predkladaná metóda nie je univerzálna: nehodí sa pre vývoj systémov, kde ide len o výber údajov, manipuláciu s nimi /napr. triedenie/ a tlač. Takisto sa nehodí pre iteračné výpočty, kde podmienka skončenia cyklu sa počíta v samotnom cykle. Prípustné sú len také cykly, kde je vopred známy /konštantný/ počet prechodov cyklom alebo podmienka skončenia cyklu je obsiahnutá vo vstupných dátach /zmena kľúča/ a cyklus slúži len na sčítanie /prípadne násobenie/ dát rovnakého typu.

Dáta vstupujúce do programu zostaveného touto metódou musia byť v určitom formáte. Preto musia byť uložené v báze dát alebo v súboroch automaticky navrhnutých touto metódou. Ak sa majú použiť iné súbory, musí sa prepracovať vstupná časť programu alebo vytvoriť program na ich konverziu.

Metóda nerieši ani tlač výstupných zostáv, na ktoré bývajú veľmi rôznorodé požiadavky - môže len vytvoriť hierarchiu výstupných súborov podporujúcu tlač zostáv.

Dáta

Dáta v systéme môžeme rozčleniť podľa viacerých kritérií.

Hlavné delenie je na vstupy a výstupy. Výstupy môžeme rozdeliť na tlačové dáta, ktoré tvoria výstupné zostavy bezprostredne použiteľné pre koncového užívateľa, a výstupy, ktoré chceme uchovať v systéme pre použitie v budúcich chodoch systému /nazveme ich pamäťové dáta/. Pamäťové dáta môžeme rozčleniť na dáta zachycujúce stav systému /nazveme ich stavové/ a dáta slúžiace na agregáciu nejakej veličiny v čase /nazveme ich akumulačné/. Príkladom stavového údaja sú denné nemocenské dávky, ktoré platia po celý čas trvania práceneschopnosti, i keby sa podklady pre výpočet dávok medzitým zmenili. Príkladom akumulačného údaja môže byť súčet mesačných miezd pracovníka za rok.

Vstupy do programu môžeme rozdeliť predovšetkým na dáta uchované v systéme z predchádzajúcich chodov, to sú stavové a akumulačné dáta, a dáta pochádzajúce z okolia systému. Dáta z okolia možno rozčleniť podľa ich časovej stálosti. Tie, ktoré sa menia od jedného chodu k druhému, sú pravé vstupné dáta. Dáta s dlhšou časovou platnosťou nazývame matričné dáta a uchovávame ich v pamäti systému. Dáta, ktorých platnosť sa dá porovnať so životnosťou systému, zabudujeme do programu ako konštanty.

Zostavenie grafu toku dát

Zostavenie grafu je podrobne popísané v /1/. Tu len stručne popíšeme hlavné zásady a potom rozdiely oproti /1/.

Vychádzame z výstupných dát. Pre každý výstupný údaj zistíme, z ktorých dát je odvodený. Ak sú to medzivýsledky, zisťujeme, z ktorých dát sú samy odvodené. Takto pokračujeme v rozklade, kým neprídeme až na vstupy. Ak niektoré z týchto vstupov sú stavové dáta, vezmeme ich ako výstupy a urobíme taký istý rozklad. Pritom zostavujeme slovník dát ako v /1/.

Rozklad môžeme znázorniť orientovaným grafom /pozri obr. 1/, kde vrcholy zodpovedajú údajom a hrany vyjadrujú vzťahy odvodenosti medzi údajmi.

Ku každému vrcholu Y priradíme operáciu O_y, ktorá ráta údaj y. Hrana (X,Y) predstavuje akúsi lokálnu pamäť typu front, kam sa ukladajú výsledky operácie O_x a odkiaľ sa vyberajú argumenty pre O_y. Počet dát, ktoré uloží O_x do frontu pri jednom vykonaní, označíme U_{XY}; počet dát, ktoré vyberie

O_y z frontu, označíme V_{XY}. Ak niektorý z týchto parametrov je >1, musíme rozlíšiť, či je konštantný alebo sa môže meniť. Napr. operácia súčtu miezd pracovníkov v oddelení potrebuje toľko sčítancov, koľko je pracovníkov, a tento počet sa môže meniť od oddelenia k oddeleniu. Parameter V súvisí s kľúčmi vo vstupných dátach, preto musíme rádove odhadnúť jeho veľkosť, aby sa mohli správne deklarovať kľúče. Ak do nejakého vrcholu vchádza hrana s V >1, predpokladáme, že je to jediná hrana, ktorá tam vchádza, tj. ide o čistú operáciu agregácie.

Každému vrcholu X v grafe priradíme prirodzené číslo T_X, pre ktoré platí: ak dva vrcholy X,Y sú spojené hranou z X do Y, potom je $T_X:T_Y = V_{XY}:U_{XY}$. Číslo T_X je úmerné počtu vykonaní operácie O_x. Ak dvom vrcholom je priradené to isté číslo T, hovoríme, že sú na rovnakej hierarchickej úrovni. V príklade na obr. 1 tieto úrovne zodpovedajú jednak hierarchii podnik - oddelenie - pracovník, ktorú nazveme organizačnou, jednak hierarchii rok - štvrťrok - mesiac, ktorú nazveme časovou. Ich kartézsky súčin dáva hierarchickú sústavu časovo-organizačných úrovní.

Dáta zoskupujeme do súborov podľa zásad uvedených v odseku Dátové súbory. Vrcholy dát z jedného vstupného alebo pamäťového súboru zlúčime do jedného vrcholu, takže každému súboru zodpovedá jeden vrchol. V ďalšom kroku zlúčime do jedného vrcholu vrcholy všetkých vstupných a pamäťových súborov na tej istej organizačnej úrovni a priradíme mu operáciu čítania všetkých týchto súborov.

Pre každý súbor s tlačovými alebo stavovými dátami pridáme ku grafu vrchol, ktorý spojíme hranami s vrcholmi, kde sa rátajú hrany tvoriace vetu tohto súboru a priradíme mu operáciu zápisu vety.

Pre graf bez stavových dát je zaručená determinovanosť výpočtu v tom zmysle, že ak zachovávame obmedzenia na poradie operácií dané grafom, pre rovnaké vstupné dáta dostávame vždy rovnaký výsledok. Ak graf obsahuje stavové dáta, musíme analyzovať precedenčné vzťahy medzi čítaním a zápisom týchto dát a podľa potreby pridať hrany na zaistenie determinovanosti /pozri odsek Zvláštne prípady/.

Vrchol ľubovoľného spojenia v grafe nazývame minimom

spojenia, ak v ňom číslo T nadobúda ostré minimum pre všetky vrcholy spojenia. Ak pre nejakú dvojicu vrcholov X,Y existuje v grafe viac spojení z X do Y a aspoň dve z nich majú rôzne minimá, potom treba obvykle vytvoriť pracovný súbor a upraviť graf /pozri odsek Zvláštne prípady/.

Algoritmus na zostavenie programu z grafu vyžaduje, aby graf obsahoval práce jeden vrchol, do ktorého nevchádza žiadna hrana. Ak je v grafe viac takýchto vrcholov, povedzme B,C,D,.., pridáme ku grafu ešte jeden vrchol, povedzme A, ktorý nazveme vstupný, a hrany (A,B),(A,C),(A,D),... Týmto hranám, napr. hrane (A,B), priradíme parametre $U_{AB}=1$, $V_{AB}=k/T_B$, kde k je najmenší spoločný násobok T_B, T_C, T_D,...

Zostavenie programu

Vstupnému vrcholu priradíme inicializáciu podmienky skončenia programu, referenčných kľúčov a pre organizačné úrovne aj pomocných agregačných premenných a čítačov dĺžky frontu. Aby sa mohli inicializovať referenčné kľúče, musia sa predtým prečítať vstupné súbory.

Pomocou nasledujúceho rekurzívneho algoritmu označíme hrany tvoriace kostru grafu.

```
proc poriadok;
  bežným vrcholom je vstupný vrchol;
  vykonať poradie;
corp
proc poradie;
  ak
    existuje pre bežný vrchol taký vrchol, že doň vchádza
    aspoň 1 neoznačená hrana h z bežného vrcholu a žiadne
    hrany z iných vrcholov
  potom
    bežným vrcholom je koniec hrany h;
    odstrániť z grafu všetky hrany vchádzajúce do bežného
    vrcholu okrem h;
  inak
    odstrániť z grafu všetky hrany vychádzajúce z bežného
    vrcholu;
```

```
      ak
        do bežného vrcholu vchádza nejaké hrana h
      potom
        bežným vrcholom je začiatok hrany h;
        označiť hranu h;
      inak
        koniec algoritmu;
      ka
    ka
    vykonať poradie;
  corp
```

Správnosť postupu pri zostavení programu z grafu toku dát je dokázaná v /2/.

Program, ktorý chceme zostaviť, má taktiež stromovú štruktúru ako kostra grafu. Označme ako F(i) podmienku, že dĺžka frontu hrany h_i, ktorá v kostre vchádza do vrcholu i, je väčšia než alebo rovná parametru V_i pre túto hranu. Označme ako strom i blok programu, v ktorom sa vykonáva podstrom kostry s koreňom vo vrchole i. Označenie i budeme používať nielen pre vrchol grafu, ale aj pre operáciu k nemu priradenú. Nech i je ľubovoľný vrchol grafu a j,...,k sú vrcholy, ktoré v kostre nasledujú za i. Potom strom i môžeme vyjadriť rekurzívne

```
strom i
  i
  ak F(j)
    strom j
  ka
  .
  .
  ak F(k)
    strom k
  ka
morta
```

V prípade, že pre hranu h_j platí $V_j=U_j=1$, odpadáva testovanie dĺžky frontu hrany h_j a

blok ak F(j) strom j ka sa zjednoduší na strom j.

Nech $V_j > 1$ je konštanta. Potom k hrane vchádzajúcej do vrcholu j priradíme čítač D_j dĺžky frontu a podmienka F(j) nadobudne tvar $D_j = V_j$. Pre organizačnú úroveň takýto čítač nulujeme prvý raz na začiatku programu; pre časovú úroveň pri založení príslušného pamäťového súboru, prípadne pri dopĺňaní súboru o ďalšie vety. Potom sa vždy nuluje tesne pred výstupom z bloku strom j. Čítač pre časovú úroveň sa ukladá a vyberá zo súboru spolu so zodpovedajúcou pomocnou agregačnou premennou.

Nech $V_k > 1$ nie je konštanta, ale závisí od zloženia vstupných dát, ako je to obvyklé pri hromadnom spracovaní dát. Vrcholu k priradíme referenčný kľúč R_k, ktorý sa porovnáva so zodpovedajúcim kľúčom K_k novo prečítanej vety. Podmienka F k má tvar $R_k \neq K_k$. Ak K_k je jednoznačný len v spojení s nadradenými kľúčmi /ktoré sú priradené vrcholom podstromu s koreňom vo vrchole k/, musíme porovnávať aj všetky nadradené referenčné kľúče so zodpovedajúcimi kľúčmi v prečítanej vete. R_k nastavujeme na hodnotu K_k na začiatku programu po čítaní prvých viet súborov a potom vždy pred výstupom z bloku strom k.

Aby sa mohli porovnávať kľúče R_k a K_k, musí byť v programe rezervované miesto pre dve vety, starú a novú /presnejšie povedané, pre dvakrát toľko viet ako je čítaných súborov/. Vždy keď sa prečíta nová veta, stará sa spracuje a porovnávajú sa kľúče referenčný a novej vety, či nedošlo k zmene kľúča.

Ako sme už spomenuli, predpokladáme, že cykly slúžia len na agregáciu /prípadne násobenie/. Pre každú agregačnú operáciu i zavedieme pomocnú agregačnú premennú PAi. Ak ide o organizačnú úroveň, prvý raz PAi nulujeme na začiatku programu; ak je to časové úroveň, prvý raz ju nulujeme pri zakladaní a dopĺňaní pamäťového súboru. Potom sa PAi nuluje vždy pred výstupom z bloku strom i. Pričítanie k PAi sa robí v programe pred vstupom do bloku strom i. Na začiatku bloku strom i sa hodnota pomocnej premennej PAi priradí príslušnej agregačnej premennej.

Ak na danej organizačnej úrovni sú nejaké akumulačné dáta

/tj. agregácie v čase/, potom môžu nastať dva prípady: buď na tejto úrovni sú aj iné dáta, ktoré sa uchovávajú v pamäti, a potom sa akumulačné dáta uchovávajú aj čítajú spolu s nimi; alebo iné dáta nie sú a vtedy sa veta pamäťového súboru číta po vstupe do prvého stromu na tejto organizačnej úrovni v programe. Vyrátané akumulačné dáta sa zapisujú do súboru pred výstupom z posledného stromu na tejto úrovni v programe.

S výnimkou inicializačnej časti /vrátane čítania prvých viet súborov/, uzavrieme program do cyklu dowhile(podmienka). Na konci cyklu vždy testujeme EOF a ak nastane, nastavíme podmienku na false.

Teraz uvedieme program zostavený podľa týchto pravidiel z grafu na obr.1. Je napísaný v pseudokóde. Premenné POi /PČi/ sú pomocné agregačné premenné pre organizačné /časové/ úrovne. print znamená zápis do tlačového súboru, write do pamäťového súboru. V texte vidno korešpondenciu vrcholov grafu s agregačnými operáciami /podľa čísel pomocných premenných, čítačov a referenčných kľúčov/ i ostatnými operáciami /podľa čísel vľavo od operácií/ okrem operácií čítania a zápisu akumulačných premenných.

```
   podmienka := true;
   PO14,PO15,PO19,PO2Ø := Ø;
   read(č.prac.,č.odd.,odprac.dni,plat.sviatky,fond prac.času,
       odmena) into nový vstup;
   read(č.prac.,č.odd.,zákl.mzda,percento,prac.kategória,daň.
       kategória,PČ11,D11,PČ12,D12) into nový matričný;
   R14,R15 := č.odd.;
   dowhile(podmienka);
     move nový vstup into starý vstup;
     move nový matričný into starý matričný;
     read(č.prac.,č.odd.,odprac.dni,plat.sviatky,fond prac.času,
         odmena) into nový vstup;
     read(č.prac.,č.odd.,zákl.mzda,percento,prac.kategória,daň.
         kategória,PČ11,D11,PČ12,D12) into nový matričný;
2    plat.dni := odprac.dni + plat.sviatky;
3    denná mzda := zákl.mzda / fond prac.času;
```

```
4    pev.zl.-prac.-mesiac := denná mzda * plat.dni;
     PO14 := PO14 + pev.zl.-prac.-mesiac;
     if (R14≠č.odd.|EOF) then
       read(č.odd.,PČ18,D18);
       pev.zl.-odd.-mesiac := PO14;
       PO19 := PO19 + pev.zl.-odd.-mesiac;
       if (EOF) then
         read(PČ23,D23,PČ27,D27,PČ24,D24,PČ28,D28);
         pev.zl.-podnik-mesiac := PO19;
         PČ23 := PČ23 + pev.zl.-podnik-mesiac;
         D23 := D23 + 1;
         if (D23=3) then
           pev.zl.-podnik-štvrťrok := PČ23;
           PČ27 := PČ27 + pev.zl.-podnik-štvrťrok;
           D27 := D27 + 1;
           if (D27=4) then
             pev.zl.-podnik-rok := PČ27;
             PČ27 := 0;
             D27 := 0;
           PČ23 := 0;
           D23 := 0;
         PO19 := 0;
       PO14 := 0;
       R14 := č.odd.;
5    prémia := pev.zl.-prac.-mesiac * percento;
6    if (prac.kategória=odmeňovaný) then
       pohyb.zl.-prac.-mesiac := odmena;
     else
       pohyb.zl.-prac.-mesiac := prémia;
7    hrubá-prac.-mesiac := pev.zl.-prac.-mesiac + pohyb.zl.-
                           prac.-mesiac;
     PČ11 := PČ11 + hrubá-prac.-mesiac;
     D11 := D11 + 1;
     if (D11=12) then
       hrubá-prac.-rok := PČ11;
       PČ11 := 0;
       D11 := 0;
```

```
8     daň-mesiac := daň hrubá-prac.-mesiac,daň.kategória ;
      PČ12 := PČ12 +daň-mesiac;
      D12 := D12 + 1;
      if (D12=12) then
        daň-rok := PČ12;
13      print(č.prac.,č.odd.,hrubá-prac.-rok,daň-rok);
        PČ12 := Ø;
        D12 := Ø;
9     čistá mzda := hrubá-prac.-mesiac - daň-mesiac;
      PO15 := PO15 + pohyb.zl.-prac.-mesiac;
      if (R15≠č.odd.|EOF) then
        pohyb.zl.-odd.-mesiac := PO15;
16      hrubá-odd.-mesiac := pev.zl.-odd.-mesiac + pohyb.zl.-
                             odd.-mesiac;
        PČ18 := PČ18 + hrubá-odd.-mesiac;
        D18 := D18 + 1;
        if (D18=12) then
          hrubá-odd.-rok := PČ18;
18        print (č.odd.,hrubá-odd.-rok);
          PČ18 := Ø;
          D18 := Ø;
        PO2Ø := PO2Ø + pohyb.zl.-odd.-mesiac;
        if (EOF) then
          pohyb.zl.-podnik-mesiac := PO2Ø;
          PČ24 := PČ24 + pohyb.zl.-podnik-mesiac;
          D24 := D24 + 1;
          if (D24=3) then
            pohyb.zl.-podnik-štvrťrok := PČ24;
            PČ28 := PČ28 + pohyb.zl.-podnik-štvrťrok;
            D28 := D28 + 1;
            if (D28=4) then
              pohyb.zl.-podnik-rok := PČ28;
30            pomer zložiek := pohyb.zl.-podnik-rok / pev.zl.-
                               podnik-rok;
29            hrubá-podnik-rok := pev.zl.-podnik-rok +
                                  pohyb.zl.-podnik-rok;
```

```
31          print(pev.zl.-podnik-rok,pohyb.zl.-podnik-rok,
                  hrubá-podnik-rok,pomer zložiek);
            PČ28 := Ø;
            D28 := Ø;
25        hrubá-podnik-štvrťrok := pev.zl.-podnik-štvrťrok +
                                   pohyb.zl.-podnik-štvrťrok;
26        print(pev.zl.-podnik-štvrťrok,pohyb.zl.-podnik-
                štvrťrok,hrubá-podnik-štvrťrok);
          PČ24 := Ø;
          D24 := Ø;
21      hrubá-podnik-mesiac := pev.zl.-podnik-mesiac +
                               pohyb.zl.-podnik-mesiac;
22      print(pev.zl.-podnik-mesiac,pohyb.zl.-podnik-mesiac,
              hrubá-podnik-mesiac);
        PO2Ø := Ø;
        write(PČ23,D23,PČ27,D27,PČ24,D24,PČ28,D28);
17    print(č.odd.,pev.zl.-odd.-mesiac,pohyb.zl.-odd.-mesiac,
            hrubá-odd.-mesiac);
      PO15 := Ø;
      R15 := č.odd.;
      write(č.odd.,PČ18,D18);
10  print(č.prac.,č.odd.,zákl.mzda,plat.dni,pev.zl.-prac.-
          mesiac,pohyb.zl.-prac.-mesiac,hrubá-prac.-mesiac,
          daň-mesiac,čistá mzda);
    write(č.prac.,č.odd.,PČ11,D11,PČ12,D12);
    if (EOF) then
      podmienka := false;
  end dowhile;
```

Dátové súbory

Tlačové dáta zoskupujeme do súborov podľa požadovaných výstupných zostáv a v rámci zostáv podľa časovo-organizačných úrovní. Vstupné dáta zoskupujeme pre každú organizačnú úroveň. Vzhľadom na efektívne využívanie externých pamätí zoskupíme pre každú organizačnú úroveň všetky maticné a pamäťové dáta.

Vety týchto súborov obsahujú kľúče navrhnuté pri zostavo-

vaní programu /premenlivé parametre V/.

V našom príklade pre výpočet miezd /pozri obr.1/ máme súbory s týmito vetami /v zátvorke vždy hierarchická úroveň/:

vstupný/pracovník/ - číslo pracovníka, číslo oddelenia, odpracované dni, platené sviatky, fond pracovného času, odmena

výplatná páska/pracovník, mesiac/ - číslo pracovníka, číslo oddelenia, mesačný plat, platené dni, pevná zložka, pohyblivá zložka, hrubá mzda, daň, čistá mzda

sumár/pracovník, rok/ - číslo pracovníka, číslo oddelenia, hrubá mzda za rok, daň za rok

sumár/oddelenie, mesiac/ - číslo oddelenia, pevná zložka, pohyblivá zložka, hrubá mzda

sumár/oddelenie, rok/ - číslo oddelenia, hrubá mzda za rok

sumár/podnik, mesiac/ - pevná zložka, pohyblivá zložka, hrubá mzda

sumár/podnik, štvrťrok/ - pevná zložka za štvrťrok, pohyblivá zložka za štvrťrok, hrubá mzda za štvrťrok

sumár/podnik, rok/ - pevná zložka za rok, pohyblivá zložka za rok, hrubá mzda za rok, pomer pohyblivej a pevnej zložky

pamäťový/pracovník/ - číslo pracovníka, číslo oddelenia, mesačný plat, daňová kategória, pracovná kategória, percento prémie, hrubá mzda za rok, čítač, daň za rok, čítač

pamäťový/oddelenie/ - číslo oddelenia, hrubá mzda za rok, čítač

pamäťový/podnik/ - pevná zložka za štvrťrok, čítač, pevná zložka za rok, čítač, pohyblivá zložka za štvrťrok, čítač, pohyblivá zložka za rok, čítač

Jednotlivé položky vo vetách a ich atribúty sú známe zo slovníka dát. Rozmery kľúčov zodpovedajú počtu rádov zodpovedajúcich parametrov V.

Ak sa používa báza dát, tieto súbory sú logickými súbormi.

Ak čítame viac súborov, musí sa pre každú možnú kombináciu hodnôt kľúčov nachádzať veta buď vo všetkých týchto súboroch alebo v žiadnom. Je to obmedzenie oproti bežným systémom na spracovanie hromadných dát, ale dá sa obísť.

Rozoberme typický prípad s jedným vstupným a jedným matričným súborom /na ktoré sa ostatné prípady dajú previesť/. Pre danú kombináciu kľúčov sú tri možnosti: veta je len vo vstupnom súbore, veta je len v matričnom súbore, vety sú v oboch súboroch.

Prvá možnosť sa vyskytuje pri aktualizácii matričného súboru /vkladanie nových záznamov/. Môžeme oddeliť aktualizáciu od samotného výpočtu ako samostatný chod, pre ktorý možno generovať jednoduchý aktualizačný program.

Druhá možnosť spravidla značí buď žiadne spracovanie alebo spracovanie so štandardnými vstupnými hodnotami. Aj jedno aj druhé sa dá ošetriť pri čítaní súborov a v programe možno generovať príslušný kód.

Zvláštne prípady

Ak na zaručenie determinovanosti grafu pridáme ďalšie hrany, tieto hrany neznamenajú, že treba ďalší operand pre operáciu priradenú k vrcholu, kam vchádza pridaná hrana, ale zaručujú len determinovanosť výpočtu. Napr. ak v bežnom mesiaci končí jedna práceneschopnosť začatá v predchádzajúcom mesiaci a začína druhá, nová práceneschopnosť, na výpočet dávok v druhej práceneschopnosti nemôžeme použiť dávky pre prvú práceneschopnosť /pozri obr.2/. Preto musíme zaručiť, že najprv sa prečítajú z pamäti systému denné dávky a vyrátajú celkové dávky pre prvú práceneschopnosť a až potom sa môžu do pamäti systému zapisovať denné dávky pre druhú práceneschopnosť. Parametre pridanej hrany (X,Y) vyrátame zo vzťahov $T_x:T_y = V_{xy}:U_{xy}$ a $\min(V_{xy},U_{xy}) = 1$. Musíme overiť, či pridaním hrany nevznikla v grafe uzavreté spojenie, čo by znamenalo, že hrana bola pridaná chybne.

Ako príklad na vytvorenie pracovného súboru môžeme uviesť výpočet podielov na hospodárskych výsledkoch, kde podiel zamestnanca na celkovej sume určenej na podiely je daný pomerom jeho ročnej mzdy k súčtu ročných miezd všetkých zamestnancov /pozri obr.3a/. Z vrcholu 5 do vrcholu 2 existujú dve spojenia. Spojenie obsahujúce vrchol 4 má v ňom minimum.

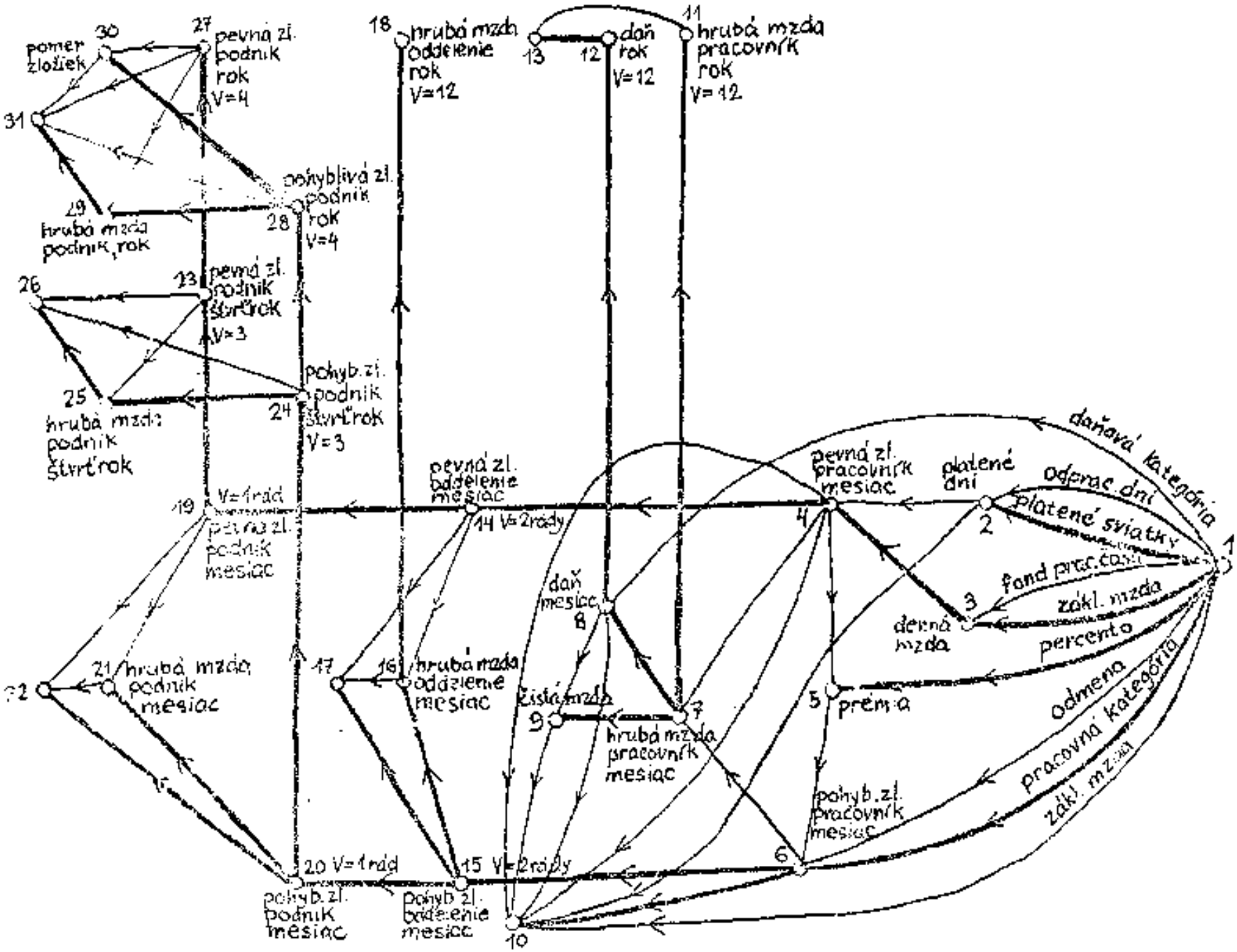
Druhé spojenie minimum nemá. Ročné mzdy pre zamestnancov sa hromadia na hrane (5,2), preto treba pre ročné mzdy zamestnancov rezervovať pracovný súbor /ak sa ročné mzdy rátajú v programe; ináč stačí súbor so mzdami previnúť a čítať znova/.

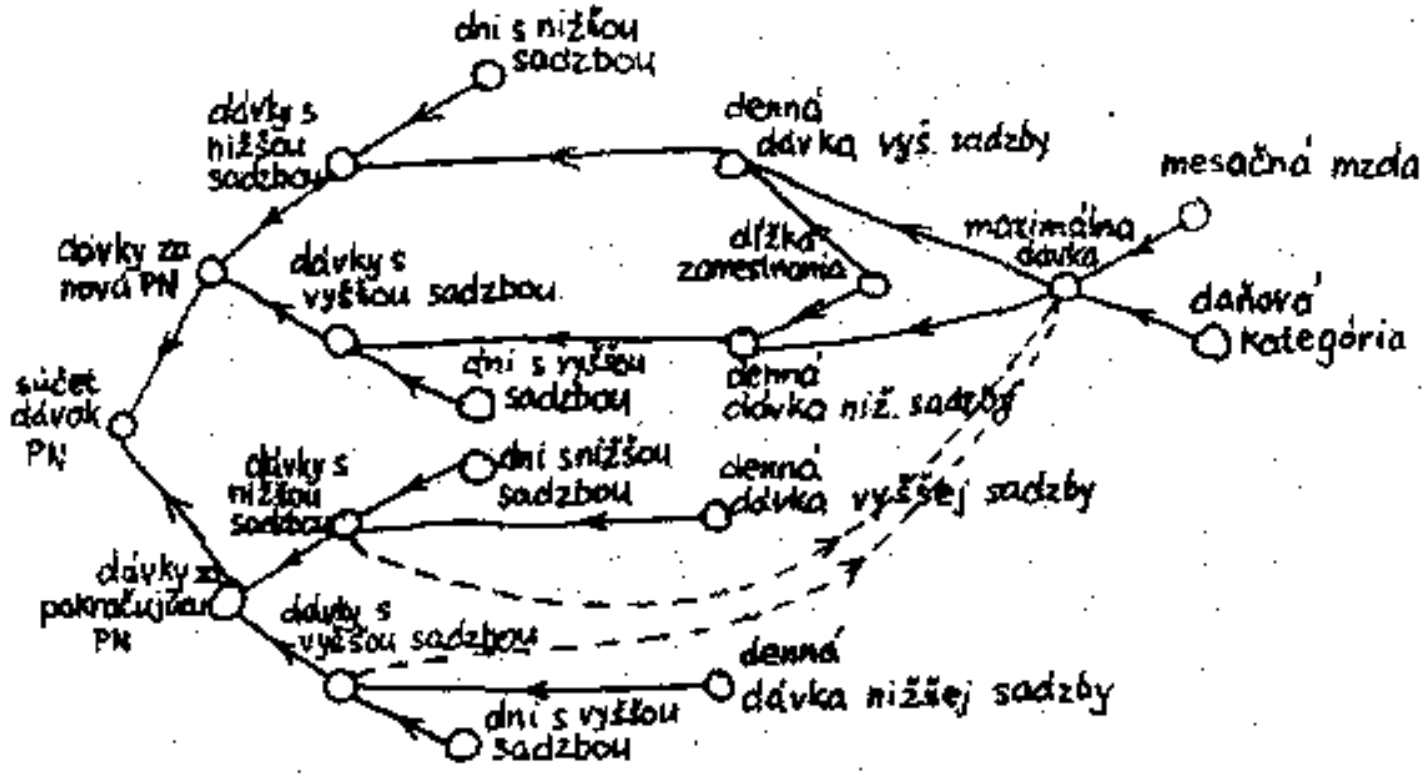
Všeobecne treba postupovať takto: ak medzi dvoma vrcholmi X, Y je viacej spojení, jedno z nich má minimum a ostatné nemajú alebo ich minimá sú väčšie, potom tieto ostatne spojenia musíme rozpojiť. Zistíme, ktorý vrchol, X alebo Y, má menšie T; nech je to Y. Potom rozdelíme Y na dva vrcholy, Y´a Y", tak, že do Y´budú vchádzať všetky hrany,ktoré pôvodne vchádzali do Y, okrem koncovej hrany spojenia s najmenším minimom, a z Y´nebude vychádzať nič. Z Y" budú vychádzať všetky hrany, ktoré pôvodne vychádzali z Y, a vchádzať do Y" bude len koncová hrana spojenia s najmenším minimom. Potom Y´predstavuje zápis do pracovného súboru a Y" čítanie z tohto súboru. V našom príklade rozdelíme vrchol 2 na vrcholy 2´a 2", kde vrchol 2´ zodpovedá zápisu ročných miezd do pracovného súboru a vrchol 2" čítaniu tohto súboru /pozri obr.3b/. Aby bola zaručená správna postupnosť vykonávania operácií, graf doplníme hranou z minima nerozpojeného spojenia do vrcholu čítania pracovného súboru, v našom príklade (4,2").

Literatúra

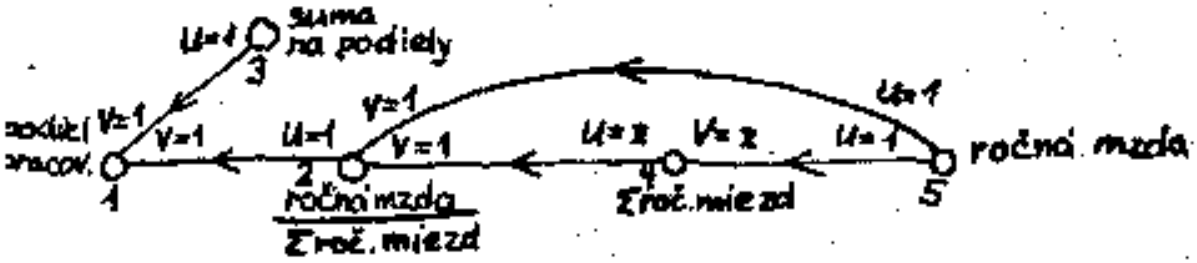
/1/ Schnapp, I.: Analýza štruktúry dát, v zborníku Programovaní '83, Dům techniky ČSVTS Ostrava, Ostrava, 1983.

/2/ Schnapp, I.a kol.: Racionalizácia využívania zdrojov počítačového systému, čiastková úloha č. 9, SPEV 905511102, VL SlPK, Bratislava, 1984


30
27
pevná zl. podnik rok V=4
pomer zložiek
31
18
hrubá mzda oddelenie rok V=12
13
12
daň rok V=12
11
hrubá mzda pracovník rok V=12
pohyblivá zl. podnik rok V=4
29
hrubá mzda podnik, rok
28
26
23
pevná zl. podnik štvrťrok V=3
25
hrubá mzda podnik štvrťrok
24
pohyb. zl. podnik štvrťrok V=3
pevná zl. oddelenie mesiac
pevná zl. pracovník mesiac
platené dni
daňová kategória
odpoč. dni
platené sviatky
19
V=1rad
pevná zl. podnik mesiac
14 V=2rady
4
2
1
fond prac. času
3
základ. mzda
denná mzda
percento
daň mesiac
8
22
21
hrubá mzda podnik mesiac
17
16
hrubá mzda oddelenie mesiac
čistá mzda
9
7
hrubá mzda pracovník mesiac
5
prémia
odmena
pracovná kategória
základ. mzda
pohyb. zl. pracovník mesiac
6
20 V=1rad
15 V=2rady
pohyb. zl. podnik mesiac
pohyb. zl. oddelenie mesiac
10



obr. 2



obr. 3a

obr. 3b