У н и в е р з и т е т у Б е о г р а д у

Математички факултет

Семинарски рад

из предмета

Методологија стручног и научног рада

Тема: НП-тешки проблеми паковања

Професор: др Владимир Филиповић

Студент: Владимир Бошковић 1134-2012

1. Увод

У овом раду најпре ће бити описане класе сложености алгоритама. Биће више речи о класама проблема *П, НП,* као и о *НП-комплетним* и *НП-тешким* проблемима. Потом следе описи *НП-тешких* проблема *паковања* као и нека њихова решења. Биће описани алгоритми за решавање једнодимензионалног и вишедимензионалног *bin-packing* проблема, као и алгоритам динамичког програмирања за решавање *проблема* 0-1 *паковања ранца.*

2. Класе сложености

Груба подела алгоритамских проблема би могла да буде на *лаке* и *тешке* проблеме, при чему би се под лаким проблемима подразумевали они проблеми за које постоје *алгоритми полиномског времена* који их исправно решавају, док за тешке проблеме такав алгоритам не постоји. За алгоритам A, временске сложености $T_A(n)$, се каже да је **алгоритам полиномског времена** ако постоји полином $p(n)$ такав да је $T_A(n) \leq p(n)$, за свако $n \in N$, тј. $T_A(n) = O(p(n^k))$, за неку константу k и величину улаза n .

Ипак, у циљу добијања финије структуре посматраних проблема, постављају се додатни услови на алгоритме који решавају проблеме и на тај начин се проблеми који задовољавају одређене критеријуме групишу у тзв. *Класе сложености.*

Класа сложеноси *П* обухвата оне алгоритамске проблеме који се могу решити у полиномском времену. У ову групу алгортиама спадају многи познати проблеми као што су алгоритми сортирања (сортирање обједињавањем, сортирање раздвајањем...), минимална повезујућа стабла, најкраћи путеви код тежинских графова итд.

Класа сложености *НП* обухвата оне проблеме чија се решења могу *верификовати (проверити)* у полиномском времену. Сваки проблем који припада класи *П,* припада

такође и класи *НП*, али је још увек нерешено питање да ли су та два скупа једнака. Заправо, питање односта класа сложености *Н* и *П* је најважнији неразјашњен проблем теоријског рачунарства. Неки од познатих проблема који припадају класи *НП* су *САТ проблем* (проблем утврђивања да ли је нека формула исказана буловским променљивама задовољива), *проблем трговачког путника* итд.

Пре саме дефиниције класа *НП-комплетних* и *НП-тешких* проблема, потребно је увести појам *редукције проблема,* односно свођења једног проблема на други. Ако су дата два проблема P_1 и P_2, P_1 се своди на P_2 ако се ефикасни алгоритам за P_2 може искористити као потпрограм у ефикасном алгоритму за P_1.

НП-комплетни **проблеми (*НП-Ц*)** представљају скуп проблема који припадају класи *НП* и који су намање „тешки“ колико и било који други проблем из класе *НП*. Заправо *НП-комплетни* проблеми представљају класу „најтежих“ *НП* проблема и уколико би постојало полиномско решење било ког *НП-комплетног* проблема, тада би сви проблеми у класи *НП* били полиномски решиви и обрнуто, уколико би се доказало да неки *НП-комплетан* проблем није решив у полиномском времену, тада ниједан *НП-комплетан* проблем не би био решив у полиномском времену. Већина научника сматра да класе проблема *НП* и *П* нису једнаке.

Понекад није јасно да ли проблем припада класи *НП,* али ипак задовољава услов да се сваки проблем из *НП* може свести на њега. Такви проблеми се називају ***НП-тешки*** **проблеми.** У наставку рада биће описани неки од *НП-тешких* проблема паковања.

3. *НП-тешки* проблеми паковања

Проблеми паковања спадају у *оптимицазионе проблеме,* односно проблеме код којих је заједничко проналажење екстремних вредности функције дефинисане на коначном скупу. Другим речима, оптимизациони проблеми се баве проналаском *оптималног решења,* где се под појмом „оптимално“ сматра „најбоље“ решење, тако да је најпре неопходно одредити меру којом се одређује квалитет неког решења.

Неки од *НП-тешких* проблема паковања су *једнодимензионални bin-packing проблем*, *вишедимензионални bin-packing* проблем, *проблем 0-1 паковања ранца, проблем паковања ранца* итд. У оквиру овог рада биће приказана нека од решења за једнодимензионални и вишедимензионални bin-packing проблем, као и за проблем 0-1 паковања ранца.

С обзиром да се ради о *НП-тешким* проблемима, не постоје ефикасни оптимални алгоритми који их решавају, већ се увек прави компромис између оптималности и ефикасности решења – неки алгоритми су ефикаснији, али имају у просеку нешто лошија решења, док код неких алгоритама важи супротно.

3.1 Једнодимензионални bin-packing проблем

Формални опис проблема гласи: дата је величина контејнера V и листа $a_1, a_2...a_n$ величина које треба спаковати у контејнере. Потребно је пронаћи цели број B, као и скупове Si (i=1,2...B), такве да је $\sum_{i \in Sk} ai \leq V$. Решење је оптимално ако је B минимално.

Постоје различити алгоритми којима се решава овај проблем, а у овом раду биће приказан алгоритам „први најбољи“ (енгл. *first-fit*).

Алгоритам не захтева да величине које треба спаковати у контејнере треба сортирати, међутим, показује се да ће алгоритам нешто боље резултате постићи уколико се оне најпре сортирају.

Поступак решавања проблема је следећи: сваку величину треба сместити у први контејнер који има довољно простора за њу. У случају да се посматрана величина не може сместити ни у један искоришћени контејнер, отворити контејнер и у њега сместити величину. Следи алгоритам који описује решење овог проблема. Претпоставка је да је капацитет контејнера већи или једнак највећој величини коју треба сместити у контејнере.

Algoritam FF.1BPP

1. $S(i) = 0$, i=1,2,...n;
2. $Bin(i) = 0$, i=1,2...n;
3. *Sortirati objekte po njihovim veličinama u nerastućem poretku*;
4. For $i = 1$ to n do
5. For $j = 1$ to n do
6. If $S(j) + a(i) \leq V$ then
7. $S(i) += a(i)$;
8. $Bin(i) = j$;
9. Break;

Низ S чува информације о „попуњености“ сваког контејнера (максимлано их може бити онолико колико има објеката које треба распоредити), па тако резултат представљеног алгоритма ће бити број елемената овог низа чија је вредност већа од 0. У низу *Bin* чувају се информације о томе где је који елемент распоређен (у који контејнер).

Основна карактеристике овог решења су једноставност и ефикасност, по цену не увек оптималног решења.

Када је реч о оптималности, треба приметити да се у било ком тренутку не може десити да два или више контејнера буду до пола или мање попуњени, односно ако је крајњи резултат B искоришћених контејнера, најмање B-1 контејнер је више од половине свог капацитета попуњен. Показује се да представљено решење не користи више од $11/9OPT$ +1 контејнера, где је OPT оптимално решење. У случају да нема почетног сортирања елемената, граница најгорег случаја износи $17/9OPT$ +2 контејнера.

3.2 Вишедимензионални bin-packing проблем

Вишедимензионални bin-packing проблем представља уопштење једнодимензионалног bin-packing проблема. Формални опис проблема гласи: нека је дато m различитих типова контејнера, где је сваки тип i (i=1,2...m) дефинисан капацитетом W_i и фиксном ценом C_i. Нека је дат скуп ствари J, величине n, где свака ствар j ($1 \leq j \leq n$) има тежину w_i. Потребно је спаковати све ствари у скуп контејнера, тако да укупна цена искоришћених контејнера буде минимална, при чему треба водити рачуна да укупна тежина ствари у једном контејнеру не прелази његов капацитет.

У оквиру овог рада биће представљена два решења – прво решење користи једнодимензионални bin-packing проблем, док се друго решење базира на *проблему збира подскупова* (енгл. *Subset problem*).

3.2.1 Решење базирано на једнодимензионалном bin-packing проблему

Резултат рада овог алгоритма јесте m изводљивих решења која представљају резултат комбиновања више различитих типова контејнера. На крају је потребно само изабрати најбоље решење међу њима.

Поступак рада алгоритма је следећи: у свакој итерацији алгоритма i (i=1,2...m), иницијализује се вредност $k = i$, као и скуп „неспакованих“ ствари $J' = J$ и одреди се скуп J_k ствари које могу бити спаковане у контејнер k (скуп ствари чији капацитет није већи од капацитета контејнера W_k). Потом је потребно решити једнодимензионални bin-packing проблем над скупом ствари J_k и контејнерима капацитета W_k. За решавање једнодимензионалног проблема, у овом решењу, користи се алгоритам из одељка 3.1. На крају итерације, потребно је ажурирати скуп J' - $J' = J' \setminus J_k$.Уколико је скуп J' празан, добијено је решење π_i, док се у супротном инкрементира вредност k (k=k+1) и понавља

поступак. Резултат алгоритма је m решења (π_1, π_2,... π_m) од којих треба изабрати најбоље (решење код којих је укупна цена искоришћених контејнера најмања).

Следи алгоритам који представља описано решење. Претпоставка је да важи $W_1 \leq W_2 \leq .. \leq W_m$ и $c_1 \leq c_2 \leq .. \leq c_m$.

Algoritam 1DIM.BPP

1. For i = 1 to m do
2. $J' = J$;
3. $k = i$;
4. $Jk = \{j \in J' \mid wj \leq Wk\}$;
5. Rešiti jednodimenzionalni bin-packing problem nad Wk i Jk;
6. $J' = J' \setminus Jk$;
7. If $J' \neq \varnothing$ then $k = k+1$; skoči na liniju 4;
8. Sačuvaj podelu π_i;
9. End For
10. Prikazati najbolju podelu;

3.2.2 Решење базирано на проблему збира подскупова

Основна идеја овог решења је следећа: за сваки тип контејнера, контејнер се пуни елементима из скупа нераспоређених ствари тако да укупна тежина распоређених ствари контејнера буде максимална, уз услов да тежина распоређених ствари не пређе капацитет контејнера. На почетку, скуп нераспоређених ствари једнак је скупу свих ствари. Потом се, за сваки тако распоређени тип контејнера, рачуна однос цене и укупне тежине распоређених ствари у том контејнеру и за коначни распоред се бира онај код кога је тај однос најмањи, док се скуп нераспоређених ствари умањује за скуп ствари смештених у тај контејнер. Поступак се понавља док се не распореде све ствари.

Следи алгоритам који представља описано решење.

Algoritam SS.BPP

1. $J' = J$;
2. For i = 1 to m do
3. $Z(i) = \textbf{\textit{Maksimizuj}} \sum_{j \in J'} w(j)$ tako da $\sum_{j \in J'} w(j) \leq W(j)$;
4. U $S(i)$ smesti stvari određene u koraku 3;
5. U i^* smestiti indeks i za koji je odnos $c(i)/Z(i)$ minimalan;
6. U kontejner označen sa i^* smestiti stvari iz $S(i^*)$ i $J' = J' \setminus S(i^*)$;
7. Ukoliko J' nije prazno, skoči na liniju 2;

3.3 Проблем 0-1 паковања ранца

Проблем 0-1 паковања ранца такође спада у *НП-тешке* проблеме паковања и основна разлика овог проблема у односу на bin-packing проблеме је у томе што нам је на располагању један ранац (контејнер), одређеног капацитета, док су ствари које треба спаковати означене тежином и „вредношћу“, па је циљ спаковати ствари тако да укупна вредност спакованих ствари буде максимална, а да укупна тежина спакованих ствари не пређе капацитет ранца. Ознака проблема „0-1“ наглашава да свака ствар мора бити узета или одбачена у целости приликом паковања ранца. Постоји и друга верзија овог проблема у којој је могуће узети део u_i ствари i чија вредност у укупном збиру учествује са $v_i*(u_i/w_i)$, где је v_i вредност ствари i и w_i тежина ствари i.

Формални опис проблема гласи: нека W капацитет ранца и нека је на располагању n ствари, нумерисаних са 1,2...n, где је њихов скуп означен са $S = \{1,2,...n\}$. Нека ствар $i \in S$ има тежину w_i и вредност v_i. Потребно је одредити вредност M тако да је

$$M = \max_{T \subseteq S}\{ \sum_{i \in T} vi \mid \sum_{i \in T} wi \leq W \}.$$

Наивно решење проблема би било генерисање свих могућих подскупова ствари и одабрати међу њима онај који има највећу цену, а чији укупни капацитет не прелази капацитет ранца. Међутим, такво решење има експоненцијалну сложеност. Применом динамичког програмирања може се добити доста бржи алгоритам, уколико је број W релативно мали.

Нека је S_i подскуп скупа S који садржи све ствари 1,2...n и нека је $S_0 = \varnothing$. Нека је „најбољи“ подскуп ствари делимичног скупа S_i онај за који се постиже максимална вредност циљне суме, при чему се масимум узима преко свих подскупова оних ствари скупа S_i чија укупна тежина не прелази

$$W - \sum_{k=i+1}^{n} wk.$$

За свако i=0,1...n и j=0,1...W, нека је $M(i,j)$ максимална укупна вредност неког подскупа делимичног скупа S_i који имају укупну тежину **тачно** једнаку j. Дакле, за i=0 $M(0,j) = 0$, за свако $j \leq W$. За i=1,

$$M(i,j) = \begin{cases} M(i-1,j), & \text{ако је } wi > j, \\ \max\{M(i-1,j), M(i-1,j-wi) + vi\}, & \text{иначе.} \end{cases}$$

С обзиром да је тражена вредност M једнака

$$M = \max M(n,j), \text{ за } j = 0,1,...W,$$

потребно је одредити вредности $M(n, j)$, за свако j = 0,1,...W и узети највеће од њих.

Следи алгоритам који представља описано решење.

Algoritam KNAPSACK(*v, w, W*)
1. For j = 0 to W do $M(j)$ = 0;
2. For i = 0 to n do
3. For j = W downto $w(i)$ do
4. If $M(j\text{-}w(i)) + v(i) > M(j)$ then
5. $M(j) = M(j - w(i)) + v(i)$;
6. M = max $M(j)$, j = 0,1,...W;
7. Return M;

Укупно време извршавања алгоритма износи $O(W) + O(n*W) + O(W)$ (прва *for* петља, друга *for* петља и линија 6). Ово време зависи од променљиве W која је део улаза, па у најгорем случају укупна сложеност овог решења може бити веома лоша (нпр. за $W=2^n$). Због тога технички овај алгоритам није ефикасан, али је у пракси овај алгоритам доста бржи од алгоритма „грубе силе“.

4. Закључак

Као што је већ речено, карактеристика *НП-тешких* проблема јесте да не постоји оптималан и ефикасан алгоритам који их решава, већ се мора правити компромис између ефикасности и оптималности решења. Идеја рада је била да се представе неки *НП-тешки* проблеми паковања и покажу нека од решења за њихово решавање. Нека решења су ефикасна, али не увек и оптимална – као што је случај са описаним алгоритмима за решавање bin-packing проблема, док су нека решења мање ефикасна (барем у теорији), али увек оптимална – као код приказаног решења проблема 0-1 паковања ранца.

Литература:

[1] Mohamed Haouari, Mehdi Serairib – "*Heuristics for the variabled sized bin-packing problem",* 2009

[2] Thomas C. Cormen, Charles E. Liesorson, Ronald L. Rivest, Clifford Stein – "*Introduction to Algorithms",* The MIT Press, 2009

[3] Dejan Živković – „*Osnove dizajna i analize algoritama*", CET, 2007

[4] *http://en.wikipedia.org/wiki/Bin_packing_problem*