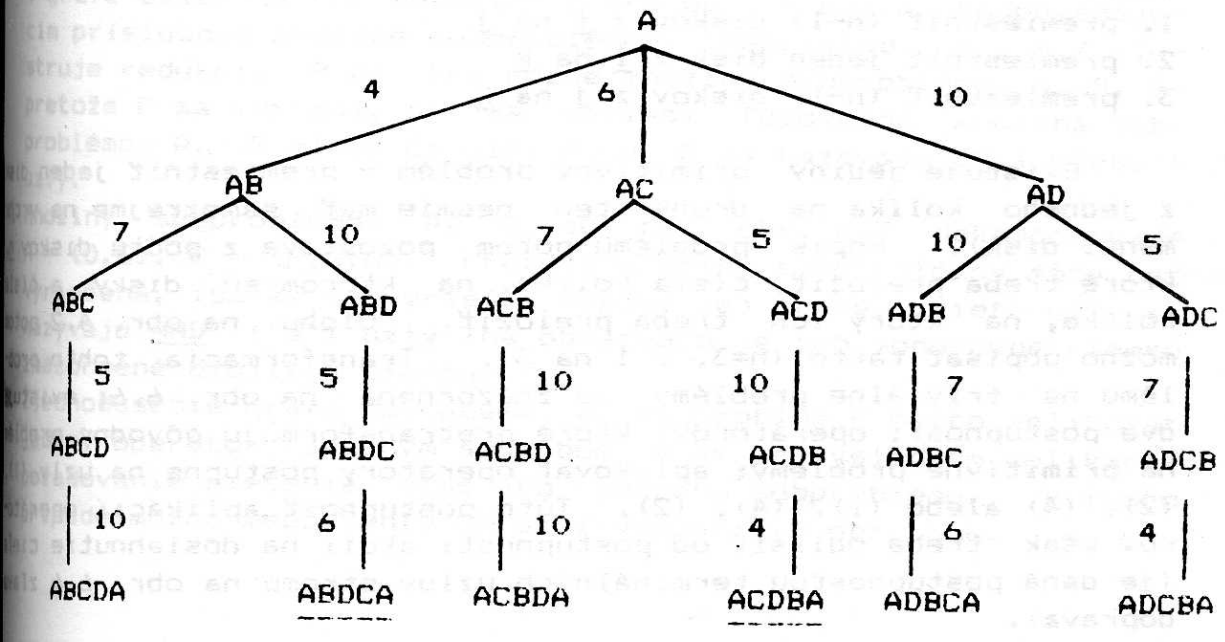
Spodné dve úrovne grafu môžeme vynechať, pretože dĺžka cesty medzi n mestami je deterministicky určená prvými (n-1) mestami.

Pretože graf stavového priestoru je zvyčajne veľmi veľký, pri hľadaní riešenia je nutné generovať iba tú časť grafu, o ktorej sa predpokladá, že obsahuje riešenie.



Obr. 6.5 Graf riešenia úlohy obchodného cestujúceho

6.1.2. *Riešenie problému redukciou problému*

Pri reprezentácii redukcie problému (úlohy) je najdôležitejšou údajovou štruktúrou opis problému (problem description), alebo ciele. Je daný opis pôvodnej úlohy, riešením je postupnosť transformácií, ktoré pretransformujú pôvodný problém na množinu subproblémov, ktorých riešenie je triviálne. Operátor môže pretransformovať jeden problém na niekoľko subproblémov, pričom na vyriešenie pôvodného problému je potrebné vyriešiť všetky subproblémy. Okrem toho, na jeden problém môže byť aplikovateľných viacero operátorov, resp. jeden operátor viacerými spôsobmi - v tomto prípade stačí riešiť jeden z možných subproblémov. Problém, ktorého riešenie je zrejmé (bezprostredné), sa nazýva primitívny (triviálny) problém (primitive problem).

Reprezentácia redukcie problému je potom definovaná trojicou:

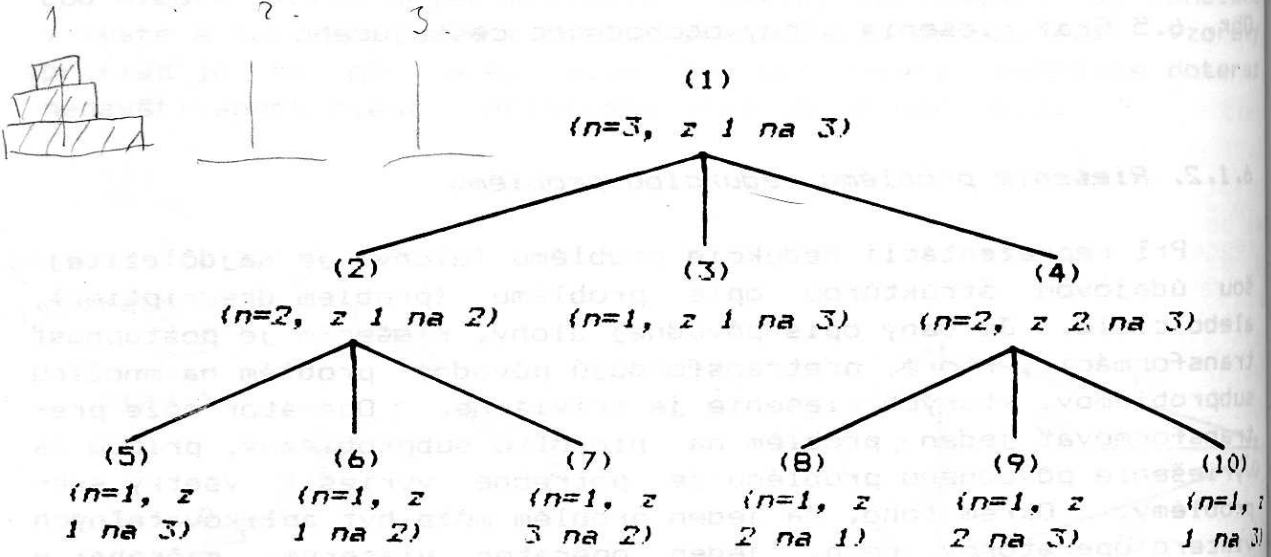
1. popis počiatočnej úlohy
2. množina operátorov pre transformáciu problému na subproblémy

3. množina opisov primitívnych problémov

Pri riešení sa využíva spätné uvažovanie. Názorným príkladom tohto typu úloh je úloha Hanojskej veže (viď aj podkapitola 4.3.2.3 a obr. 4.2). Pri riešení potrebujeme iba jeden operátor: ak sú dané 3 kolíky i, j, k; potom možno problém premiestniť n>1 diskov z kolíka i na kolík k pretransformovať na tri subproblémy:

1. premiestniť (n-1) diskov z i na j
2. premiestniť jeden disk z i na k
3. premiestniť (n-1) diskov z j na k

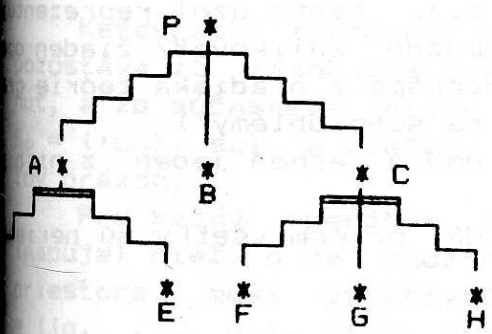
Existuje jediný primitívny problém - premiestniť jeden disk z jedného kolíka na druhý (ten nesmie mať samozrejme na vrchu menší disk). Popis problému potom pozostáva z počtu diskov n, ktoré treba preložiť, čísla kolíka, na ktorom sú disky, a čísla kolíka, na ktorý ich treba preložiť. Úlohu na obr. 4.2 potom možno popísať takto (n=3, z 1 na 3). Transformácia tohto problému na triviálne problémy je znázornená na obr. 6.6; existujú dve postupnosti operátorov, ktoré pretransformujú pôvodný problém na primitívne problémy: aplikovať operátory postupne na uzly (1), (2), (4) alebo (1), (4), (2). Túto postupnosť aplikácií operátorov však treba odlíšiť od postupnosti akcií na dosiahnutie cieľa (je daná postupnosťou terminálnych uzlov stromu na obr. 6.6 zľava doprava).



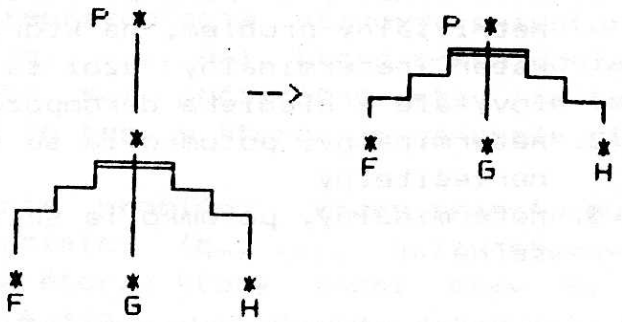
Obr. 6.6 Riešenie Hanojskej veže redukciou problému

AND/OR grafy. Zovšeobecnením stromu pre riešenie redukciou problému dostaneme AND/OR grafy, ktoré sú konštruované podľa týchto pravidiel [1]:

1. Každý uzol reprezentuje buď jediný problém alebo množinu problémov, ktoré treba vyriešiť. Koreňový uzol zodpovedá pôvodnému problému.
2. Uzol, ktorý zodpovedá primitívnemu problému, je terminálnym uzlom, ktorý nemá potomkov.
3. Ak operátor transformuje problém P na množinu subproblémov, v grafe existuje orientovaná hrana od P k uzlom, zodpovedajúcim príslušnej množine subproblémov. Napríklad obr. 6.7 ilustruje redukciu P na tri rôzne množiny subproblémov A, B, C- pretože P sa vyrieši, ak sa vyrieši ľubovoľná množina subproblémov A, B alebo C; uzly A, B, C sa nazývajú OR ("alebo") uzly.
4. Množiny subproblémov A, C sú na obr. 6.7 dekomponované A = {D,E}, C = {F,G,H}. Pretože množina subproblémov môže byť vyriešená, iba ak sa vyriešia všetky jej prvky, tieto uzly sa nazývajú AND ("a") uzly (na obrázkoch je ich vodorovné rameno znázornené dvojitou čiarou).
5. Zjednodušenie grafu dostaneme, ak na problém P možno aplikovať jediný operátor jediným spôsobom a ak je výsledkom aplikácie požadovanie riešenia viac ako jedného subproblému. V tomto prípade možno medziľahlý OR uzol vypustiť - obr. 6.8.



r. 6.7 AND/OR graf

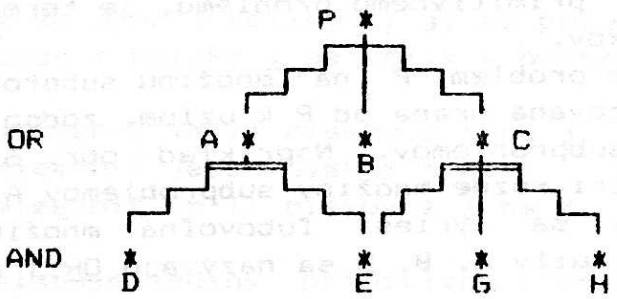


Obr. 6.8 Zjednodušenie AND/OR grafu

Na obr. 6.7, 6.8 reprezentuje každý uzol rôznu úlohu alebo ožinu úloh. Keďže ku každému uzlu (okrem koreňového) existuje áve jeden rodičovský uzol, dané grafy sú vlastne AND/OR stromy. ažujme variáciu obr. 6.7, úloha A je dekomponovateľná na podohy D, E a úloha C na E, G, H. Potom môže byť úloha E reprentovaná buď dvoma rôznymi uzlami alebo jedným uzlom - obr. 6.9. toho, akú reprezentáciu zvolíme, závisí aj algoritmus prehľavania.

Graf riešenia (solution graph), príp. strom, riešenia je ograf, ktorého koreňový uzol je riešiteľný, čo je postačujuce

pre nájdenie riešenia pôvodnej úlohy (doterajšie úvahy sa týkali grafov zahrňujúcich celý priestor prehľadávania).



Obr. 6.9 AND/OR graf

Uzol je riešiteľný, ak je:

1. terminálny (triviálny problém)
2. neterminálny, jeho potomkovia sú uzly typu AND, pričom sú všetky riešiteľné
3. neterminálny, jeho potomkovia sú uzly typu OR a aspoň jeden z nich je riešiteľný

Podobne, uzol je neriešiteľný, ak je:

1. neterminálny a nemá potomkov - t.j. tento uzol reprezentuje netriviálny problém, na ktorý nemožno aplikovať žiaden operátor (neterminálny uzol sa tu nechápe z hľadiska teórie grafov, ale z hľadiska dekompozície na subproblémy!)
2. neterminálny, potomkovia sú typu AND a aspoň jeden z nich je neriešiteľný
3. neterminálny, potomkovia sú typu OR, pričom všetky sú neriešiteľné

Vzťah medzi redukciou úloh a reprezentáciou stavovým priestorom: aj keď je niektorá reprezentácia pre daný problém prirodzenejšia, je možné pretransformovať problém do druhej formy. Napríklad úlohu Hanojskej veže možno riešiť prehľadávaním stavového priestoru použitím operátorov, ktoré premiestnia jeden disk. V porovnaní s reprezentáciou pomocou redukcie problému, ktorá v skutočnosti dáva priamo algoritmus riešenia, môže byť prehľadávanie stavového priestoru veľmi náročné.

Prevod stavového priestoru na reprezentáciu redukciou problému - sú možné dva prístupy. V prvom sa graf stavového priestoru chápe ako AND/OR graf, obsahujúci iba uzly typu OR. Každý stav stavového priestoru zodpovedá úlohe dostať sa z daného stavu do cieľového stavu; cieľovému stavu stavového priestoru zodpovedá triviálna úloha dostať sa z cieľového do cieľového stavu. Údajo-

vé štruktúry sa nemenia, mení sa iba ich interpretácia; problém je daný stavovou informáciou a implicitným cieľom.

V druhom prístupe je potrebné predefinovať operátory stavového priestoru. Každý operátor, prevádzajúci stav i na stav j, sa zmení na operátor, aplikovateľný na problém, ako sa dostať zo stavu i do cieľového stavu. Tým dochádza k redukcii problému na dvojicu subproblémov:

1. choď zo stavu i do stavu j (triviálny problém)
2. choď zo stavu j do cieľového stavu

Prevod redukcie problému na reprezentáciu stavovým priestorom - tento prevod je trochu obtiažnejší (dané existenciou AND uzlov). Pri riešení problému pomocou redukcie problému môžeme o počiatočnom probléme predpokladať, že je zložený z dvoch komponent:

1. opis cieľa, ktorý treba dosiahnuť - q_0
2. opis počiatočného stavu sveta - s_0

Napríklad:

a. q_0 - teoréma, ktorú treba dokázať
   s_0 - axiómy, na základe ktorých treba túto teorému dokázať
b. q_0 - cieľová konfigurácia objektov
   s_0 - existujúca konfigurácia

Každý stav zodpovedajúcej reprezentácie stavovým priestorom pozostáva zo zásobníka cieľov (q_1, ..., q_0), ktorý treba dosiahnuť, a zo súčasného stavu sveta s_0; počiatočný stav je teda $S_0 = ((q_0), s_0)$ a cieľový stav je ten, v ktorom je zásobník cieľov prázdny.

Pre každý operátor redukcie problému, ktorý transformuje (mapuje) cieľ q na množinu podcieľov $\{q_m, \ldots, q_n\}$ v stavovom priestore, musí existovať operátor, ktorý zmení stav $S_1 = ((q_1, \ldots, q_0), s)$ na stav $S_2 = ((q_m, \ldots, q_n, q_1, \ldots, q_0), s)$ - množina cieľov $(q_m, \ldots, q_n)$ sa pridá na vrchol zásobníka cieľov (v poradí, v akom by sa mali vykonať, ak je to dôležité) a stav sveta s sa nezmení.

V stavovom priestore potrebujeme ešte jeden typ operátorov, ktoré možno aplikovať, ak sa na vrchole zásobníka nachádza triviálny problém - z vrcholu zásobníka sa odstráni triviálny problém a zodpovedajúcim spôsobom sa zmení stav sveta s. Pri riešení Hanojskej veže zachycuje nový stav sveta zmenenú pozíciu jedného disku. Pri dokazovaní teorém sa nový stav líši od starého pridaním jednej formuly k tým, ktoré boli zadané ako axiómy, resp. boli pridané po vyriešení predchádzajúcich subproblémov. Reprezentácia tohto typu je použitá v systéme STRIPS [1].

c. Ak x už bol v S-CLOSED, neurob nič (aj keď bola nájdená nová cesta do x, jej cena nemôže byť menšia ako cena už nájdenej cesty).

d. Choď na (2).

4. Zo zoznamu T-OPEN vyber uzol n, pre ktorý je gt(n) minimálne a vlož ho do T-CLOSED. Ak je n aj v S-CLOSED, choď na (5). Inak pre každého predchodcu x uzla n urob:

a. Ak x nie je ani v T-OPEN ani v T-CLOSED, vlož ho do T-OPEN. Pripoj smerník z x na n a polož gt(x) = gt(n)+c(x,n).

b. Ak x už bol v T-OPEN a bola nájdená kratšia cesta z x do t, uprav podľa toho hodnotu gt(x) a nastav smerník z x na n.

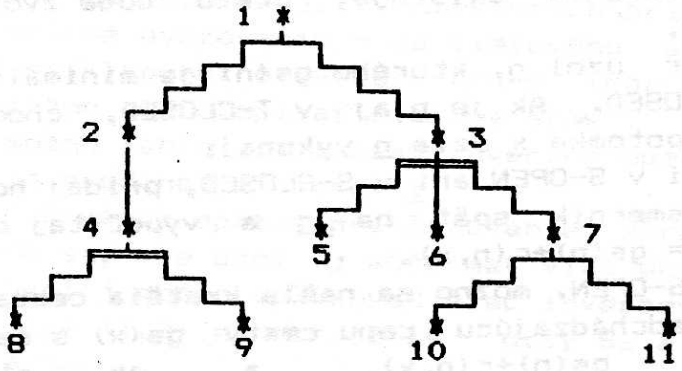
c. Ak x už bol v T-CLOSED, neurob nič.

d. Choď na (2).

5. Uvažuj množinu uzlov, ktoré sú v S-CLOSED a aj v T-CLOSED alebo T-OPEN. Z tejto množiny vyber uzol n, pre ktorý je gs(n)+gt(n) minimálne. Riešením je cesta z n späť do s a dopredu do t.

6.2.2. Slepé prehľadávanie AND/OR grafov

V tomto prípade môže byť problém definovaný špecifikovaním počiatočného uzla (reprezentuje pôvodný, počiatočný problém), množinou terminálnych uzlov (reprezentujú triviálne problémy) a množinou operátorov pre redukciu cieľa na podciele. Riešenie pôvodného problému je dané podgrafom, ktorý postačuje na vyriešenie počiatočného uzla. AND/OR graf na obr. 6.15 (uzly 5, 6, 8, 9, 10, 11 sú terminálne uzly) má tri možné riešenia – tri subgrafy riešenia sú {1,2,4,8,9}, {1,3,5,6,7,10}, {1,3,5,6,7,11}.



Obr. 6.15 AND/OR graf

Expandovanie uzla je trochu odlišné, ako v prípade stavového priestoru. Napríklad na uzol 1 na obr. 6.15 možno aplikovať dva operátory, jeden ho redukuje na jediný ekvivalentný problém (uzol 2) a druhý na množinu troch subproblémov (uzly 5, 6, 7). V tomto prípade by boli uzly 2, 3, 5, 6, 7 generované pri expandovaní uzla 1 a každému z nich by bol priradený smerník na svojho predchodcu, ale iba uzly 2, 5, 6, 7 by boli vložené do zoznamu neexpandovaných uzlov. Ďalej uvedený algoritmus využíva spätné uvažovanie (od počiatočného problému) a vychádza z dvoch predpokladov:

1. priestor prehľadávania je AND/OR stromm, a nie všeobecný AND/OR graf
2. ak sa problém transformuje na množinu subproblémov, tie môžu byť riešené v ľubovoľnom poradí

Z prvej podmienky vyplýva, že ten istý subproblém sa môže vyskytnúť v rôznych uzloch stromu, pričom stále sa musí riešiť znovu.

Prehľadávanie AND/OR grafu do šírky - uvedený algoritmus nájde strom riešenia (ak riešenie existuje) minimálnej hĺbky (medziľahlé OR uzly sú ignorované pri výpočte hĺbky stromu). O počiatočnom uzle sa predpokladá, že nie je terminálnym.

Algoritmus:

1. Vlož počiatočný (štartovací) uzol do zoznamu OPEN neexpandovaných uzlov.
2. Vyber prvý uzol n zo zoznamu OPEN.
3. Expanduj uzol n a vygeneruj všetkých jeho potomkov; pre každého jeho potomka m, ak m reprezentuje množinu viac ako jedného subproblému, vygeneruj potomkov m. Ku každému vygenerovanému uzlu pripoj smerník na jeho predchodcu (otca). Všetky nové uzly, ktoré zatiaľ nemajú potomkov, vlož na koniec zoznamu OPEN.
4. Ak neboli v kroku (3) vygenerovaní žiadni potomkovia, potom:
   a. Označ uzol n ako neriešiteľný.
   b. Ak neriešiteľnosť n spôsobí neriešiteľnosť niektorých jeho predkov, označ ich ako neriešiteľné.
   c. Ak je štartovací uzol neriešiteľný, algoritmus končí neúspechom.
   d. Odstráň z OPEN všetky uzly, ktoré majú neriešiteľného predka.
5. Ak boli v kroku (3) vygenerované nejaké terminálne uzly, potom:
   a. Označ tieto terminálne uzly ako vyriešené.
   b. Ak vyriešenie týchto terminálnych uzlov umožňuje vyriešenie niektorých ich predkov, označ ich ako vyriešené.
   c. Ak je štartovací uzol označený ako vyriešený, algoritmus končí úspechom.

d. Odstráň z OPEN všetky uzly, ktoré sú vyriešené, alebo ich predchodca je vyriešený.

6. Choď na (2).

Prehľadávanie AND/OR grafu do hĺbky - oproti predchádzajúcemu algoritmu stačí zmeniť krok (3) takto:

3' Ak je hĺbka uzla n menšia ako zvolená maximálna hĺbka, potom: Expanduj uzol n - vygeneruj všetkých jeho potomkov a pre každého potomka m, ak m reprezentuje množinu viac ako jedného subproblému, vygeneruj potomkov m (podľa charakteru daných subproblémov). Ku každému vygenerovanému uzlu pripoj smerník na jeho predchodcu (otca). Všetky nové uzly, ktoré zatiaľ nemajú potomkov, vlož na začiatok zoznamu OPEN.

Tento algoritmus nájde riešenie, ak existuje v rámci zvolenej maximálnej hĺbky; pre výpočet hĺbky na koniec kroku (3) možno ešte pridať:

Pre každý uzol x, ktorý sa pridáva do OPEN, vypočítaj jeho hĺbku ako hĺbku uzla n plus 1.

Ak je hĺbka počiatočného uzla 0, potom hĺbka ľubovoľného uzla x je dĺžka sekvencie operátorov, ktoré musíme aplikovať, aby sme sa dostali do uzla x.

6.2.3. Heuristické prehľadávanie stavového priestoru

Predpokladajme, že definície počiatočných stavov, operátorov a cieľových stavov sú dané. Otázkou je, ako prehľadávať takto definovaný stavový priestor efektívne. To si vyžaduje existenciu dodatočnej informácie o vlastnostiach danej problémovej oblasti (mimo tej, ktorá je obsiahnutá v definíciách stavov a operátorov). Informácia tohto druhu sa nazýva heuristická informácia a príslušné metódy heuristické metódy prehľadávania.

Väčšinu heuristických techník prehľadávania možno študovať ako variáciu metód slepého prehľadávania pre ten istý typ reprezentácie problému.

Heuristická informácia môže byť využitá pri prehľadávaní vo viacerých bodoch:

1. pri rozhodovaní, ktorý uzol expandovať ako ďalší
2. pri expandovaní uzla rozhodnúť - ktorého potomka alebo potomkov generovať - miesto "slepého" generovania všetkých možných potomkov
3. pri rozhodovaní, či dané uzly možno zanedbať alebo odseknúť zo stromu prehľadávania

Ďalej bude uvedený algoritmus, ktorý využíva heuristickú informáciu iba podľa 1. bodu, s tým, že daný uzol sa expanduje úplne alebo vôbec nie. Ako prvý sa vždy expanduje uzol, ktorý je "najsľubnejší". Prehľadávanie, realizované na tomto princípe, sa

2. Aj keď cena riešenia je dôležitá, kombinatorická zložitosť problému môže byť taká veľká, že prípustný A* algoritmus nemôže dobehnúť do konca. Možno získať na rýchlosti za cenu určitého zhoršenia kvality riešenia?
3. Niekedy je ťažké nájsť dobrú heuristickú funkciu, spĺňajúcu podmienku prípustnosti; slabá, ale prípustná heuristická funkcia h* degeneruje A* na slepé prehľadávanie. Ako je prehľadávanie ovplyvnené voľbou neprípustnej heuristickej funkcie?

4.2.5. *Heuristické prehľadávanie AND/OR grafov*

Hlavný rozdiel oproti prehľadávaniu stavového priestoru spočíva v existencii AND uzlov, ktoré sťažujú prehľadávanie. Každý uzol AND/OR grafu reprezentuje cieľ, ktorý treba dosiahnuť. Budeme predpokladať spätné uvažovanie od počiatočného uzla (koreňa) k množine podcieľov. AND/OR grafy sú v tomto zmysle použité na reprezentáciu redukcie problému (rozdiel oproti stavovému priestoru spočíva aj v tom, že operátory stavového priestoru majú jeden vstup a jeden výstup a operátory redukcie problému môžu mať jeden vstup a niekoľko výstupov).

Môžeme uvažovať aj prehľadávanie AND/OR grafov v priamom smere od triviálnych problémov k problému, ktorý chceme vyriešiť (cieľ) - typický problém pri dokazovaní teorém.

Definícia optimálneho riešenia - cena stromu riešenia môže byť definovaná dvoma spôsobmi:

1. Celková cena stromu riešenia je daná ako suma cien (váh) všetkých hrán stromu.
2. Maximálna cena stromu riešenia je suma cien pozdĺž najdrahšej cesty z koreňa do terminálneho uzla.

Ak je cena každej hrany 1, celková cena stromu sa rovná počtu hrán stromu a maximálna cena hĺbke najhlbšieho uzla.

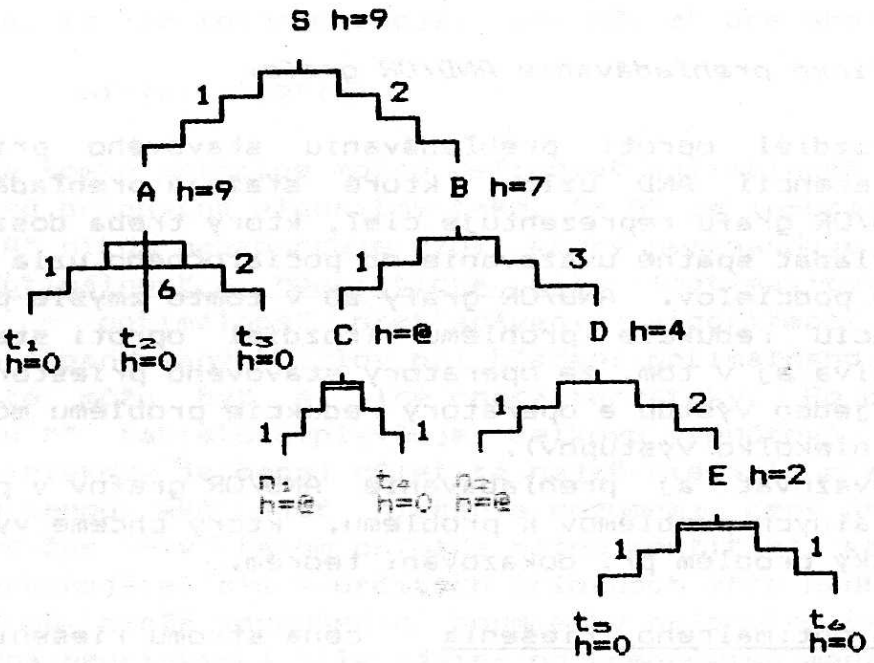
Nech c(n,m) je cena hrany z uzla n do uzla m. Definujme funkciu h(n) takto:

1. Ak je n terminálny uzol (triviálny problém), potom h(n)=0.
2. Ak má n OR potomkov, potom h(n) je minimum cez všetkých potomkov m z c(n,m)+h(m).
3. Ak má n AND potomkov a využíva sa celková cena, potom h(n) je suma cez všetkých potomkov m z c(n,m)+h(m).
4. Ak má n AND potomkov a využíva sa maximálna cena, potom h(n) je maximum cez všetkých potomkov m z c(n,m)+h(m).
5. Ak je n neterminálny uzol (netriviálny problém) a nemá potomkov, potom h(n) je nekonečno.

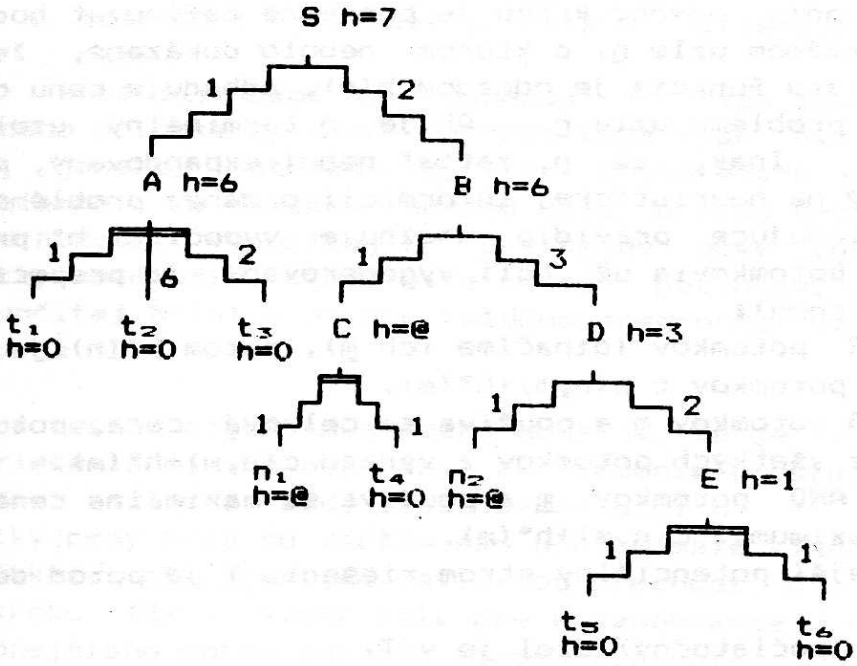
Je zrejmé, že h(n) je konečné, iba ak je problém reprezen-

tovaný uzlom n riešiteľný. Pre každý riešiteľný uzol n, udáva h(n) cenu optimálneho stromu riešenia daného problému.

Uvažujme AND/OR graf podľa obr. 6.16, označené sú ceny hrán aj hodnoty celkovej ceny stromu. Uzly, ktoré nemajú potomkov, sú označené t (terminálny, triviálny problém) alebo n (neriešiteľný). Optimálne riešenie zodpovedá podgrafu, obsahujúcemu uzly S, B, D, E, t_5, t_6. Ak použijeme maximálnu cenu - obr. 6.17, optimálne riešenie zodpovedá podgrafu S, A, t_1, t_2, t_3.

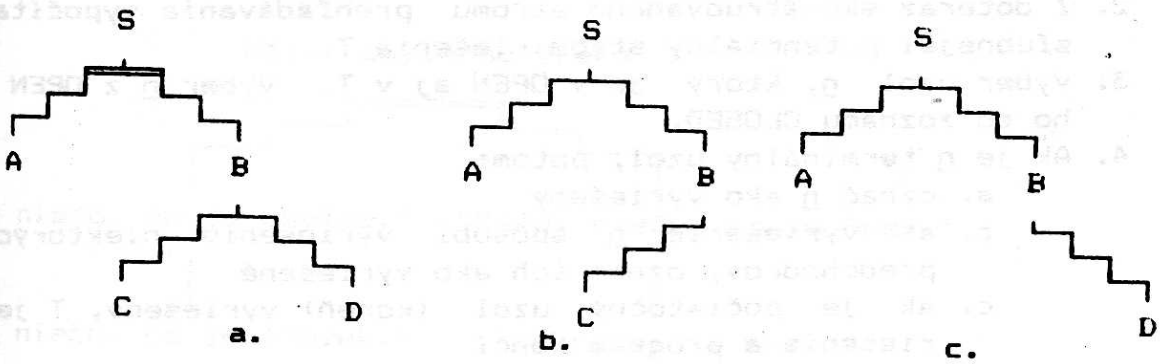


Obr. 6.16 Celková cena



Obr. 6.17 Maximálna cena

Algoritmus usporiadaného prehľadávania AND/OR grafu - pri prehľadávaní AND/OR grafov neexistuje jednojednoznačná korešpondencia medzi voľbou uzla na expandovanie a voľbou potenciálneho riešenia. Uvažujme napr. AND/OR graf na obr. 6.18a. Pretože uzly C, D sú typu OR, expandovaním uzla B dostaneme dva potenciálne stromy riešenia - obr. 6.18b, c.



Obr. 6.18 a. AND/OR graf b.- c. Potenciálne stromy riešenia

Nilssonov algoritmus - jednotlivé uzly sa vyberajú na expandovanie v dvoch krokoch:

1. identifikovať najsľubnejší potenciálny strom riešenia
2. zvoliť uzol tohto stromu pre expandovanie

Pre vykonanie prvého kroku je potrebné definovať hodnotiacu funkciu h* v každom uzle n, o ktorom nebolo dokázané, že je neriešiteľný. Táto funkcia je odhadom h(n), odhaduje cenu optimálneho riešenia problému uzla n. Ak je n terminálny uzol, potom h*(n)=h(n)=0. Inak, ak n zatiaľ nebol expandovaný, musí byť odhad založený na heuristickej informácii o danej problémovej oblasti. Nasledujúce pravidlo umožňuje vypočítať h* pre každý uzol, ktorého potomkovia už boli vygenerovaní (a prepočítať pri expandovaní stromu):

1. Ak má n OR potomkov (označíme ich m), potom h*(n) je minimum cez týchto potomkov z c(n,m)+h*(m).
2. Ak má n AND potomkov m a používa sa celková cena, potom h*(n) je suma cez všetkých potomkov z výrazu c(n,m)+h*(m).
3. Ak má n AND potomkov m a používa sa maximálna cena, potom h*(n) je maximum z c(n,m)+h*(m).

Najsľubnejší potenciálny strom riešenia T je potom definovaný takto:

1. Štartovací (počiatočný) uzol je v T.
2. Ak strom prehľadávania (doteraz vygenerovaná časť priestoru prehľadávania) obsahuje uzol n a AND potomkov uzla n, potom všetci títo potomkovia sú v T.
3. Ak strom prehľadávania obsahuje uzol n a OR potomkov uzla n, potom v T je ten z potomkov uzla n, pre ktorý c(n,m)+h*(m) je minimálne (cez všetky m).

Algoritmus usporiadaného prehľadávania AND/OR stromu môže byť definovaný takto:

1. Vlož štartovací uzol s do zoznamu neexpandovaných uzlov OPEN.
2. Z doteraz skonštruovaného stromu prehľadávania vypočítaj najsľubnejší potenciálny strom riešenia T.
3. Vyber uzol n, ktorý je v OPEN aj v T. Vyber n z OPEN a vlož ho do zoznamu CLOSED.
4. Ak je n terminálny uzol, potom:
   a. označ n ako vyriešený
   b. ak vyriešenie n spôsobí vyriešenie niektorých jeho predchodcov, označ ich ako vyriešené
   c. ak je počiatočný uzol (koreň) vyriešený, T je strom riešenia a program končí
   d. odstráň z OPEN všetky uzly, ktorých predkovia sú vyriešení
5. Ak uzol n nemá potomkov (t.j. nemôže byť aplikovaný žiaden operátor), potom:
   a. označ uzol n ako neriešiteľný
   b. ak neriešiteľnosť n spôsobí neriešiteľnosť niektorých jeho potomkov, označ ich ako neriešiteľné
   c. ak je štartovací uzol neriešiteľný, program končí neúspechom

d. odstráň z OPEN všetky uzly s neriešiteľnými predchodcami

6. Inak expanduj uzol n, generuj všetkých jeho potomkov, pre každého potomka m, ktorý reprezentuje viac ako jeden subproblém, generuj potomkov m, ktorí zodpovedajú jednotlivým subproblémom. Ku každému novému uzlu pripoj smerník na svojho predchodcu a vypočítaj h* pre každý nový vygenerovaný uzol. Všetky nové uzly, ktoré ešte nemajú potomkov, zaraď do OPEN. Prepočítaj h*(n) a h* pre každého predchodcu uzla n.
7. Choď na (2).

Pre tento algoritmus možno dokázať, že je prípustný - nájde strom riešenia minimálnej ceny, ak riešenie existuje a ak:
a. $h^*(n) \leq h(n)$ pre každý expandovaný uzol n
b. všetky ceny hrán sú väčšie ako určité malé kladné číslo d.

Efektívnosť algoritmu závisí od presnosti h* a od implementácie kroku (3) - výber uzla pre expandovanie v rámci nájdeného najsľubnejšieho potenciálneho stromu riešenia.

Vzájomne závislé subproblémy (interdependent subproblems)- doteraz sa predpokladalo, že všetky subproblémy môžu byť riešené nezávisle, takže riešenie jedného nemá vplyv na riešenie druhého. Táto podmienka je často porušená, najmä pri:
a. úlohách, vyžadujúcich konzistentné naviazanie premenných
b. problémoch s obmedzenými zdrojmi

Pre ilustráciu prvého typu úloh uvažujme úlohu: "Dokážte, že existuje omylný Grék."; celý priestor prehľadávania je na obr. 6.19.

Obr. 6.19 AND/OR graf vyžadujúci konzistentné naviazanie premennej "niečo"

Algoritmus Nilssonovho typu tu neuspeje z dvoch dôvodov:

1. Nemá mechanizmus, ktorý by objavil, že "Turing je človek." a "Sokrates je Grék." nemôže byť riešením.
2. Ak by aj taký mechanizmus existoval, algoritmus nemá prostriedky pre zmenu riešenia, ktoré sa už našlo. Ak "Turing je človek." je prvý triviálny problém, ktorý sa našiel, potom "Nájdi niečo, čo je človek." a "Nájdi niečo, čo je omylné." sa označia ako vyriešené a preto "Sokrates je človek." sa odstráni zo zoznamu OPEN a "Nájdi niečo, čo je Grék." sa tým pádom, použitím predchádzajúcej hodnoty "niečo" (tá je naviazaná na "Turing") stane neriešiteľné.

Príklad druhého typu úloh: "Ukážte, že Tom vie zviesť herečku. Zvedenie herečky možno zredukovať na získanie auta a získanie jachty. Tom má 5000$ a auto stojí 5000$ a jachta tiež 5000$.". Nilssonov algoritmus by chybne odpovedal, že Tom môže zviesť herečku. Problémy tohto typu sa vyskytujú aj pri plánovaní činnosti robota. Pre riešenie úloh tohto typu boli navrhnuté zovšeobecnené AND/OR grafy (Levi, Sirovich, 1975, 1976) [1], v ktorých môžu mať redukčné operátory dva alebo viac vstupných uzlov.

6.2.6. Prehľadávanie stromov hier

Procedúra minimax – uvažujme strom na obr. 6.20, každý uzol reprezentuje pozíciu hry, neterminálne uzly sú označené menom hráča, ktorý je na ťahu. Celý strom je konštruovaný z pohľadu hráča A a úlohou je nájsť jeho najlepší ťah v pozícii 1. Terminálne uzly sú označené hodnotou pre hráča A – *V*, *P*, *R* (výhra, prehra, remíza).