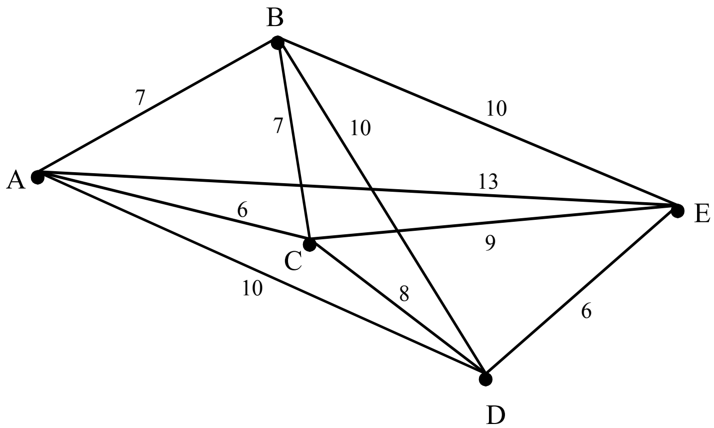
Zadatak 7: Problem trgovačkog putnika

Trgovački putnik mora da poseti svaki od pet gradova prikazanih na slici 18. Između svakog para gradova postoji put, dužine naznačene na slici. Polazaći od grada A, naći minimalan put koji obezbeđuje posetu svakom gradu samo jedanput i povratak u A. Predložiti dve različite heurističke funkcije. Za svaku od funkcija primenom nekog od algoritama pretraživanja naći rešenje problema. Koja od predloženih funkcija daje bolje rešenje?



Slika 18

Rešenje

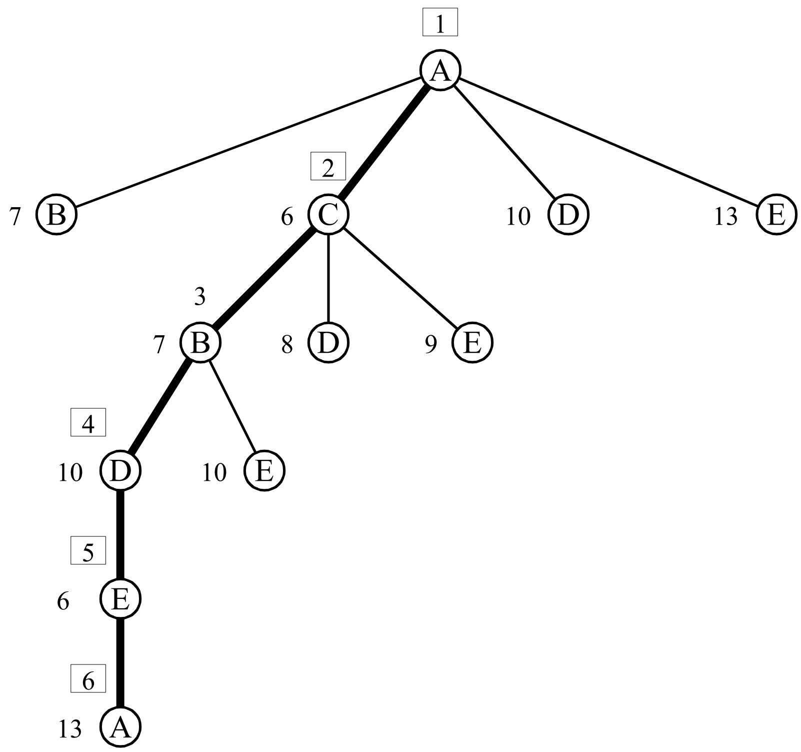
Najjednostavnija je heuristika da u svakom koraku pretraživanja zadatog grafa prioritet damo jednom od neobiđenih gradova koji je najbliži tekućem gradu (i do koga postoji put iz tekućeg grada). Funkcija koja odgovara ovoj heuristici definiše se na sledeći način:

Vrednost heurističke funkcije h_1 čvora Y - naslednika tekućeg čvora X - jednaka je dužini puta između gradova X i Y.

$h_1(Y) = \textit{rastojanje}(X,Y)$

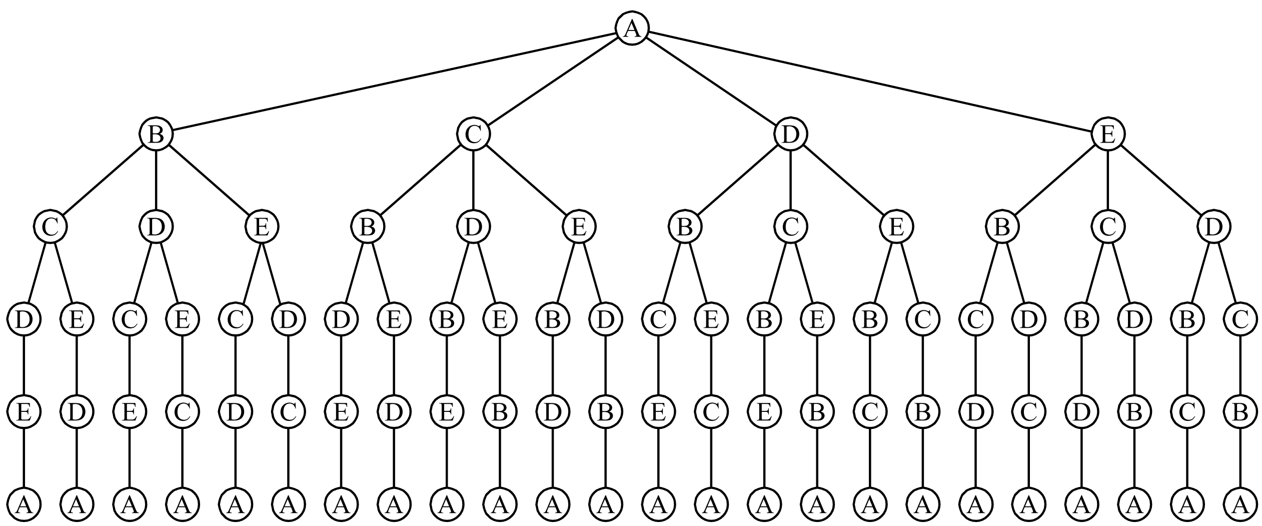
Pošto nam heuristička funkcija služi da ocenimo čvor lokalno među sledbenicima tekučeg čvora, prirodno joj odgovara algoritam planinarenja. Stablo pretrage za dati problem koje se dobija primenom planinarenja prikazano je na slici 19.

Put A-C-B-D-E-A koji se dobija kao rešenje je dužine 42 i nije minimalan, jer planinarenje i ne garantuje nalaženje minimalnog rešenja. Minimalan put za dati problem je A-C-D-E-B-A (ili ista ruta u suprotnom smeru) koji ima dužinu 37.



Slika 19

Da bismo dobili garantovano minimalno rešenje, moramo ga tražiti primenom algoritma A*. Heuristička funkcija h_1 nije pogodna u ovom slučaju jer je vrednost funkcije h_1 za proizvoljan čvor stabla pretrage već uračunata u cenu parcijalne putanje do tog čvora, tako da bi se algoritam A* sveo na algoritam 'granaj i ograniči' što bi imalo za posledicu otkrivanje gotovo kompletnog stabla pretrage za dati problem. Kompletno stablo pretrage prikazano je na slici 20.



Slika 20

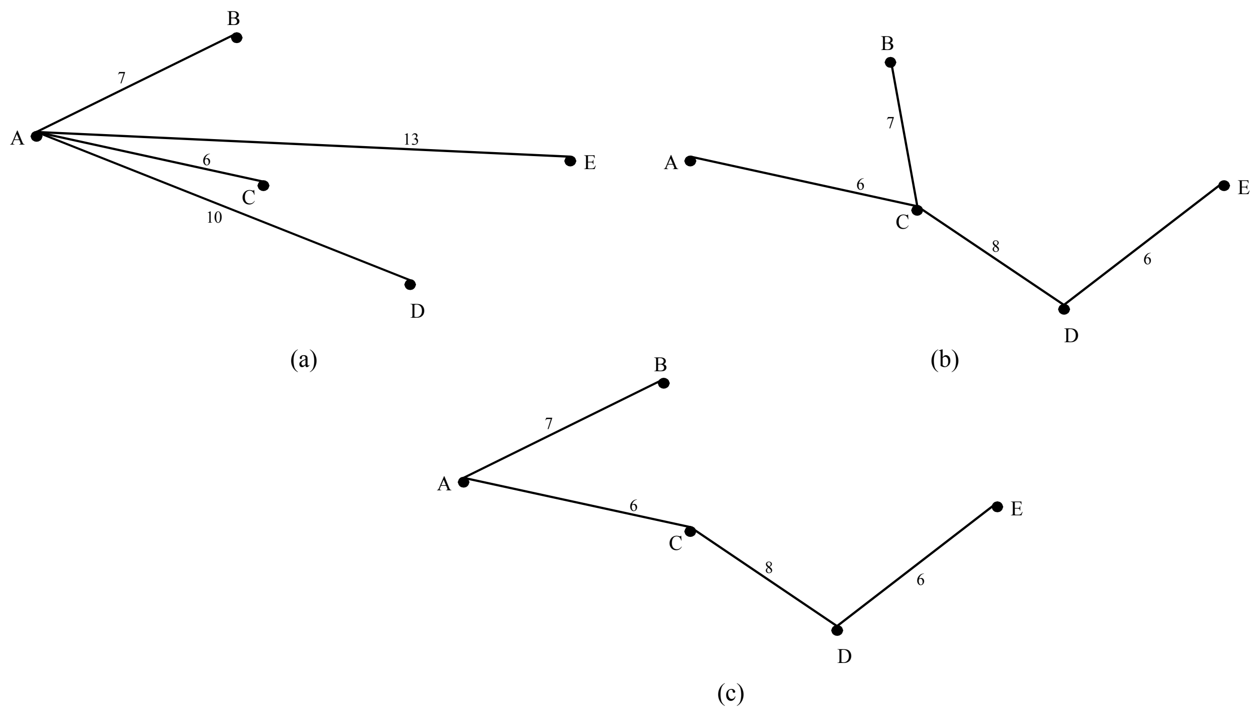
Razmotrimo heurističku funkciju h_2:

Vrednost heurističke funkcije h_2 čvora Y na parcijalnoj putanji P od korena stabla pretrage jednaka je dužini takozvanog *minimalnog razapinjućeg stabla* (MRS) koje obuhvata sve čvorove grafa G koji se ne nalaze na putanji P osim čvorova Y i A:

$$h_2(Y) = \text{dužina MRS}((G \backslash P) \cup \{A, Y\})$$

Razapinjuće stablo u nekom grafu je skup grana grafa koje povezuju sve čvorove grafa tako da nije formirana nijedna zatvorena petlja. Dužina razapinjućeg stabla je zbir dužina svih grana koje ga sačinjavaju. Minimalno razapinjuće stablo u grafu ima najmanju dužinu od svih razapinjućih stabala tog grafa.

Slika 21 prikazuje tri različita razapinjuća stabla koja odgovaraju kompletnom grafu pretrage sa slike 18. Razapinjuća stabla sa slike 21*b* i 21*c* ujedno predstavljaju dva minimalna razapinjuća stabla za ovaj graf.



Slika 21

Dužina minimalnog razapinjućeg stabla za dati graf može poslužiti kao procena dužine puta koji mora preći trgovački putnik. Pokazuje se da je u pitanju uvek potcenjena vrednost. U našem slučaju dužina minimalnog razapinjućeg stabla je 27, dok je dužina minimalnog puta za trgovačkog putnika jednaka 37. Funkcija h_2 koristi dužinu minimalnog razapinjućeg stabla neobiđenog dela grafa pretrage za procenu preostalog dela puta. S obzirom da funkcija h_2 uvek daje potcenjenu vrednost, postoji garancija da će algoritam A* uvek dati optimalno rešenje.

Za zadati graf G, minimalno razapinjuće stablo H određuje se na osnovu sledećeg algoritma:

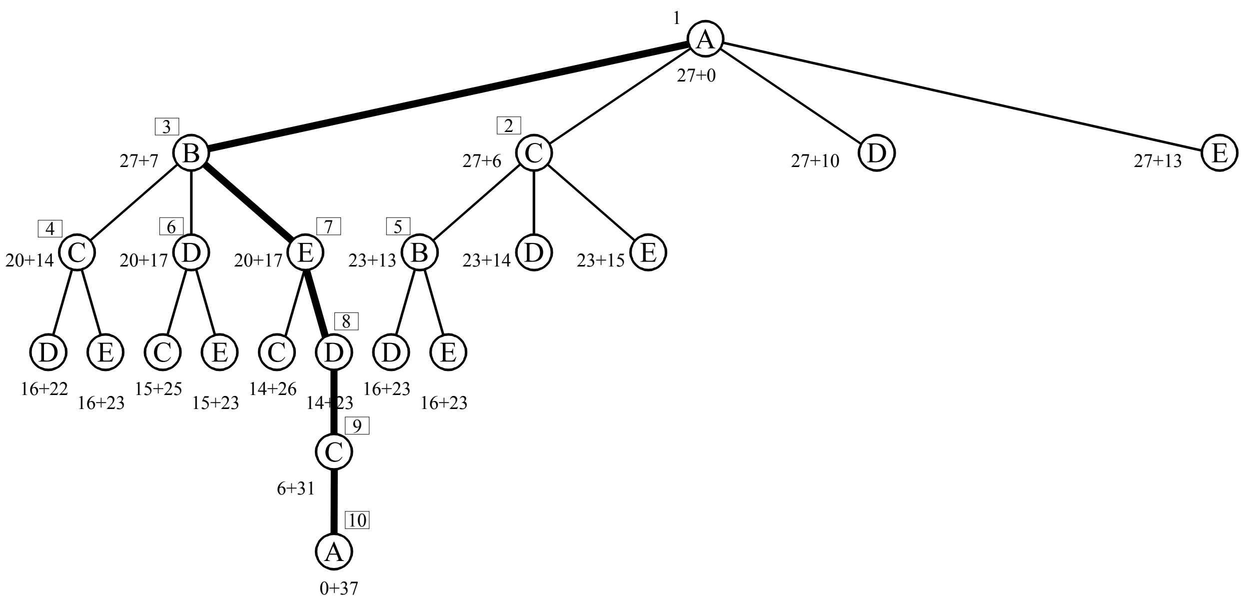
1. Inicijalno stablo H sadrži sve čvorove grafa G i nijednu granu.

2. U stablo H uključujemo najkraću granu grafa G koja nije u H, a koja sa granama iz H ne zatvara konturu.

3. Ponavljamo korak 2. dok se ne formira jedinstveno stablo H koje povezuje sve čvorove.

Za graf na slici 18, primenjujući ovaj algoritam biramo redom grane AC, DE, BC (posle toga ne može AB, jer bi se formirala kontura AB, BC, AC) i CD i tako dobijamo razapinjuće stablo sa slike 21*b*. Ukoliko posle grana AC i DE izaberemo granu AB, tada ne može BC već se bira AB tako da se dobija stablo sa slike 21*c*.

Stablo pretrage korišćenjem algoritma A* i heurističke funkcije h_2 prikazano je na slici 22. U pretrazi, od dva čvora sa jednakim funkcijama procene, prioritet je dat čvoru sa manjom vrednošću heurističke funkcije.

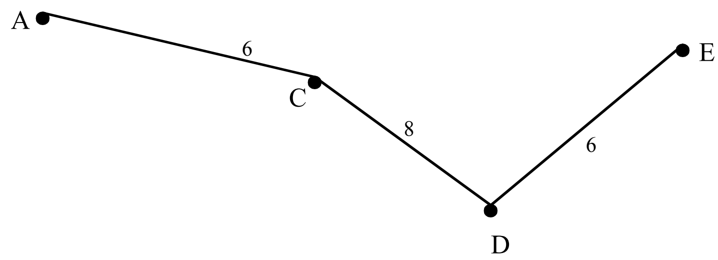


Slika 22

Vrednost funkcije procene f prikazana je uz svaki čvor stabla pretrage na uobičajen način kao zbir vrednosti heurističke funkcije (levi sabirak) i funkcije cene parcijalne putanje (u ovom slučaju dužina puta) od korena stabla pretrage do datog čvora (desni sabirak). Kao primer određivanja funkcije procene tokom pretrage razmotrimo vrednost funkcije f u čvoru E, na parcijalnoj putanji A-B-E:

- h_2(E)= dužina MRS({A,C,D,E})

  Primetimo da se dužina minimalnog razapinjućeg stabla određuje za deo polaznog grafa iz koga je udaljen čvor B (i grane koje ga povezuju sa drugim čvorovima) jer se ovaj čvor već nalazi na parcijalnoj putanji, dok su u grafu ostali čvorovi A i E s obzirom da moramo proceniti preostali put od čvora E do čvora A preko čvorova C i D. Minimalno razapinjuće stablo za traženi podgraf je prikazano na slici 23 i njegova dužina iznosi 20, koliko je na slici 23 i naznačeno.



Slika 23

- Dužina parcijalne putanje c(A-B-E) = rastojanje(A,B) + rastojanje(B,E) = 7 + 10 = 17 tako da je ukupna vrednost funkcije f za čvor E jednaka 37.

Diskusija

Pri rešavanju problema trgovačkog putnika algoritmom A* ne sme se koristiti princip dinamičkog programiranja koji nalaže da se od svih parcijalnih putanja do određenog rada razmatra samo najkraća od njih. Da je ovaj princip korišćen, u stablu pretrage sa slike 22 eliminisao bi se, na primer, čvor E na parcijalnoj putanji A-C-D-E jer je dužina te parcijalne putanje jednaka 14 i veća od dužine parcijalne putanje A-D koja je jednaka 10. Parcijalna putanja A-C-D se nalazi na ciljnoj putanji, tako da bi se eliminacijom te putanje eliminisalo i minimalno rešenje. Kod problema trgovačkog putnika moraju se, prema tome, ravnopravno razmatrati sve parcijalne putanje do određenog čvora.

U literaturi se mogu naći i druge heuristike za rešavanje problema trgovačkog putnika koje nisu zasnovane na algoritmima pretraživanja i koje u praksi daju dobre rezultate.

Zadatak 8: Problem zamenjivanja brojeva

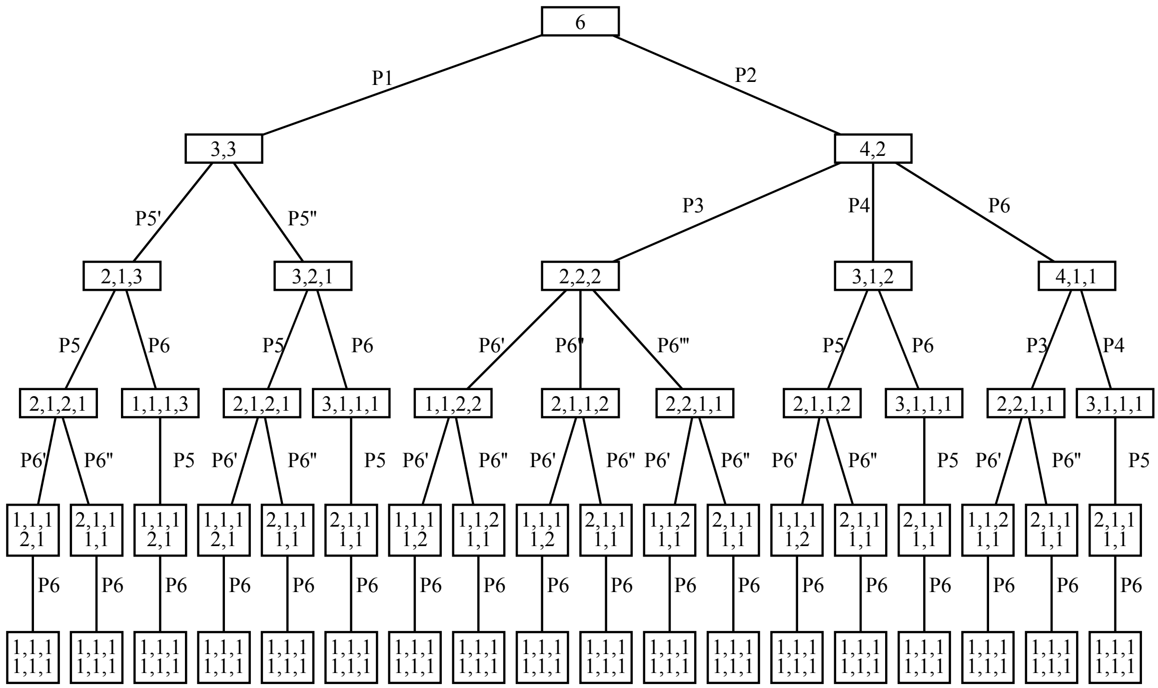
Data su sledeća pravila koja se mogu iskoristiti da zamene brojevi na levoj strani nizom brojeva na desnoj strani:

$6 \rightarrow 3,3$	$4 \rightarrow 2,2$	$3 \rightarrow 2,1$
$6 \rightarrow 4,2$	$4 \rightarrow 3,1$	$2 \rightarrow 1,1$

Kako se mogu iskoristiti ova pravila da se broj 6 transformiše u niz jedinica? Pokazati kako AO* algoritam obavlja ovu transformaciju. Usvojiti da je cena k-konektora k jedinica, a vrednost heurističke funkcije h u čvoru označenom brojem 1 je nula a čvora označenog sa n iznosi n.

Analiza problema

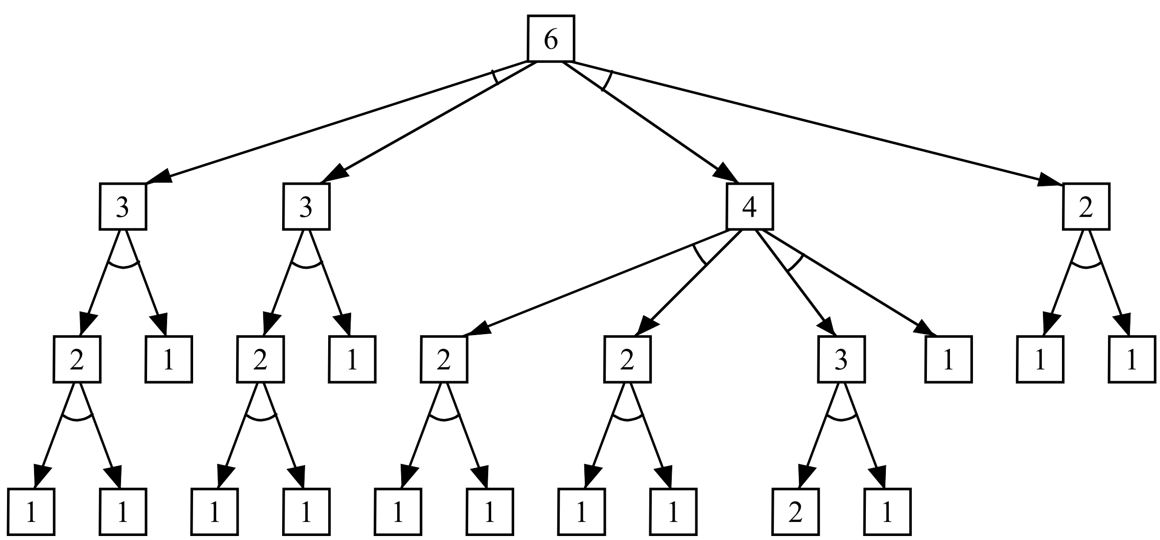
Rešenje postavljenog problema mogli bismo potražiti klasičnim algoritmima pretrage. Kompletno stablo pretrage u tom slučaju prikazano je na slici 24. Ovo stablo sadrži dosta redundanse. Na primer, za rešenje problema nije bitno da li u stanju koje je opisano listom 2,1,3 prvo zamenjujemo cifru 2 primenom pravila P6 pa onda cifru 3 primenom pravila P5 ili obrnuto, jer je u oba slučaja kranji rezultat isti.



Slika 24

Zadati problem može se *razložiti* (dekomponovati) na niz potproblema tako da svaki problem rešavamo nezavisno. Način na koji zamenjujemo pojedinu cifru jedinicama možemo rešavati nezavisno od zamene ostalih cifara u istom stanju. Na primer, da bismo rešili problem prelaska iz stanja 4,2 u ciljno stanje 1,1,1,1,1,1 posebno ćemo posmatrati problem zamene cifre 4 jedinicama i problem zamene cifre 2 jedinicama.

Pogodna predstava razloženog problema zamene cifara jedinicama je uz upotrebu AND-OR stabla. Za zadati problem AND-OR stablo predstavljeno je na slici 25. Čvorovi stabla su pojedinačni potproblemi, u ovom slučaju pojedinačne cifre. Čvorovi su povezani takozvanim *k-konektorima*. Radi se o generalizovanim granama koje povezuju jedan čvor-roditelj sa k čvorova naslednika.



Slika 25

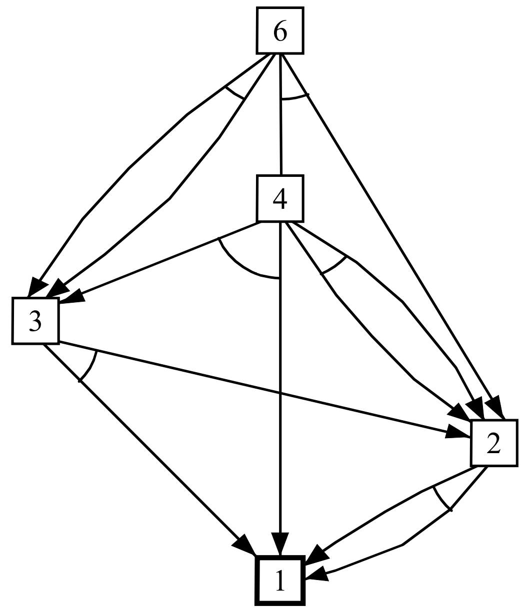
Konektor je predstavljen nizom grana koje čvor roditelj povezuju sa svakim od naslednika i koje su sve međusobno povezane lukom. Na primer, iz korenog čvora 6 polaze dva 2-konektora:

- Levi 2-konektor povezuje čvor 6 sa dva čvora, oba označena cifrom 3. Ovaj konektor izražava pravilo P1 da se cifra 6 može zameniti sa dve cifre 3.
- Desni 2-konektor povezuje čvor 6 sa čvorovima 4 i 2. Ovaj konektor izražava pravilo P2 da se cifra 6 može zameniti ciframa 4 i 2.

Prema tome, problem zamene cifre 6 jedinicama može se rešiti ILI zamenom 6 sa 3 I 3 ILI zamenom 6 sa 4 I 2. U opštem slučaju, potproblem je rešen ako je rešen primenom bilo kog od konektora iz odgovarajućeg čvora grafa, pri čemu po tom konektoru svi čvorovi naslednici moraju biti rešeni. Konektor dakle, izražava I relaciju (engl. AND) dok postojanje više konektora iz istog čvora izražava ILI relaciju (engl. OR) pa su po tome AND-OR stabla i dobila ime.

Jedna terminološka napomena: U nekim problemima pretrage AND-OR stabla imaju osobinu da iz svakog čvora grafa ide ili jedan k-konektor (k>1) ili k 1-konektora. Čvorovi za koje važi prvo svojstvo nazivaju se tada AND čvorovi, a čvorovi sa drugim svojstvom OR čvorovi. Primer AND čvora u stablu sa slike **Error! Bookmark not defined.** bio bi čvor 3. U istom stablu, međutim, koreni čvor 6 nije ni AND čvor ni OR čvor.

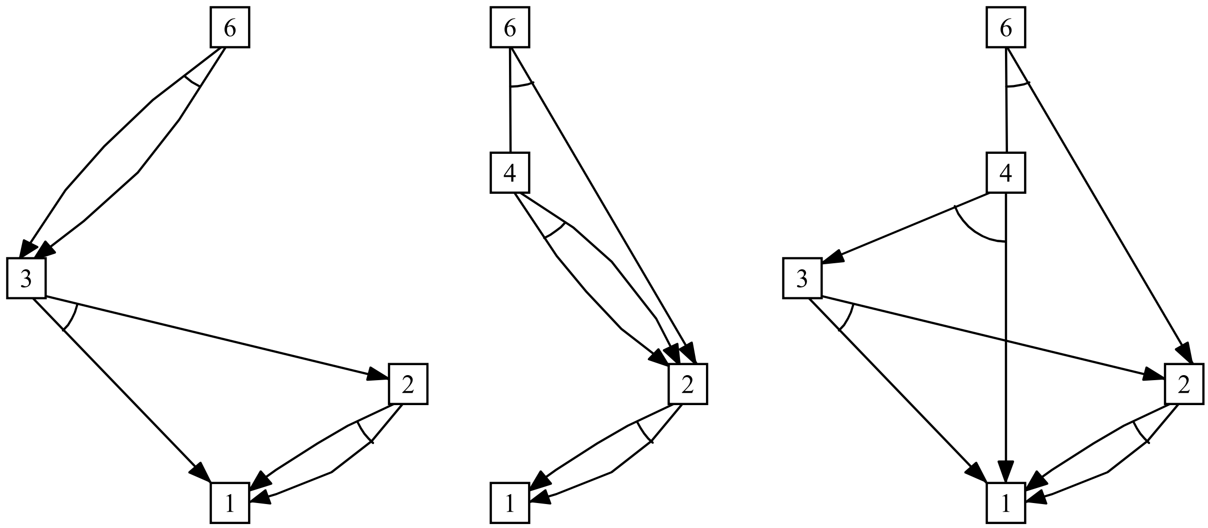
U zadatom problemu može se primetiti da će isto rešenje za zamenu određene cifre biti primenljivo nezavisno od stanja u kome se nalazi ta cifra. Na primer, rešenje koje dobijemo za cifru 4 biće primenljivo i u stanju 4,1,1 kao i u stanju 4,2. Koristeći ovo svojstvo možemo rešenje problema predstaviti još kompaktnije koristeći, umesto AND-OR stabla, AND-OR aciklički graf. AND-OR aciklički graf (u nastavku ćemo ga skraćeno nazivati AND-OR grafom) je vrsta acikličkog grafa kod koga su grane generalizovane k-konektorima. U literaturi se ovakvi grafovi ponekad nazivaju i hipergrafovima. Za zadati problem AND-OR graf predstavljen je na slici 26. Početni čvor pretrage za nalaženje rešenja je čvor 6, a ciljni čvor je 1 jer sve cifre treba zameniti jedinicama.



Slika 26

Kod 'klasičnih' metoda pretrage, rešenje je predstavljeno putanjom u grafu pretrage od početnog do nekog od ciljnih čvorova. Pri korišćenju AND/OR grafova, cilj se predstavlja (u opštem slučaju) skupom ciljnih čvorova *N*. Rešenje je predstavljeno podgrafom G' kompletnog grafa pretrage G. Rešenje se, ako postoji, dobija tako što se, polazeći od startnog čvora *n*, izabere jedan od konektora koji od čvora *n* vodi ka čvorovima-naslednicima n_1, n_2,..., n_k. Ukoliko svaki od čvorova naslednika predstavlja ciljni čvor (dakle jedan od čvorova iz skupa ciljnih čvorova *N*), rešenje je pronađeno i sastoji se od izabranih čvorova povezanih izabranim konektorom. U suprotnom slučaju, za svaki od čvorova naslednika koji nije ciljni čvor, potrebno je izabrati jedan od konektora i uključiti taj konektor i njegove čvorove-naslednike u rešenje. Procedura uključivanja novih konektora i čvorova u rešenje ponavlja se sve dok u podgrafu G' postoji čvor koji nije ciljni, a za koji nije izabran konektor.

S obzirom da procedura određivanja rešenja u AND/OR grafu uključuje proizvoljan izbor konektora, u opštem slučaju postoji više rešenja određenog problema pretrage. Razmotrimo nalaženje rešenja za dati problem na osnovu AND/OR grafa sa slike 26. Za dati problem postoje tri različita rešenja prikazana na slici 27. Rešenje dobijamo polazeći od čvora 6. Moguće je izabrati jedan od dva konektora koji predstavljaju pravila P1 i P2. Ukoliko izaberemo levi konektor koji odgovara pravilu P1, u podgraf rešenja uključujemo startni čvor, izabrani konektor i čvor 3 koji predstavlja jedinog naslednika čvora 6 po konektoru P1. S obzirom da čvor 3 nije rešenje problema (da se podsetimo, jedini ciljni čvor je čvor 1), potrebno je izabrati izlazni konektor iz čvora 3. Radi se o jednom jedinom konektoru koji predstavlja pravilo P5. U podgraf rešenja uključujemo izabrani konektor P5 i čvorove-naslednike čvora 3 po konektoru P5, a to su čvorovi 2 i 1. Sada je potrebno za čvor 2 izabrati jedini izlazni konektor P6 i dodati taj konektor u rešenje čime se dobija kompletno rešenje prikazano na slici 27*a*. Ukoliko se u prvom koraku nalaženja rešenja izabere alternativni izlazni konektor čvora 6, moguće je dobiti druga dva moguća rešenja datog problema prikazana na slici 27*b* i *c*.



(a) (b) (c)

Slika 27

Konektorima u AND/OR grafu, mogu se pridružiti cene koje reprezentuju cene upotrebe određenih pravila pri rešavanju problema. Na osnovu ovih cena može se definisati cena određenog rešenja čime se ustanovljava kriterijum za poređenje različitih rešenja. Cena k(n,N) za neki podgraf G' grafa G od startnog čvora n do skupa ciljnih čvorova N definiše se sledećom rekurzivnom formulom:

- Ako je n element skupa N, onda je $k(n, N) = 0$.
- Inače, čvor n poseduje izlazni konektor prema skupu čvorova $n_1, n_2, ..., n_i$. Neka je cena ovog konektora c_n. Tada je cena k(n , N) kompletnog rešenja jednaka zbiru cena izlaznog konektora čvora n i cena svih podgrafova od čvorova naslednika do ciljnih čvorova iz skupa N:

$$k(n,N) = c_n + k(n_1 , N) + k(n_2 , N) + ... + k(n_i , N)$$

Odredimo cenu rešenja sa slike 27*a* prema ovoj definiciji:

$$k(6,\{1\}) = k(P1) + k(3,\{1\}) + k(3,\{1\})$$

Kada se zamene cene podgrafova iz pojedinih međučvorova koje iznose

$$k(3,\{1\}) = k(P5) + k(2,\{1\}) + k(1,\{1\})$$

$$k(2,\{1\}) = k(P6) + k(1,\{1\}) + k(1,\{1\})$$

$$k(1,\{1\}) = 0$$

dobija se

$$k(6,\{1\}) = k(P1) + 2*k(P5) + 2*k(P6) = 10$$

s obzirom da je cena svakog od pravila jednaka 2.

Treba primetiti da se u ceni rešenja cene pojedinih konektora uračunavaju više puta. U opštem slučaju, ukoliko postoji m različitih putanja u grafu rešenja od startnog čvora do nekog čvora *n*, u cenu rešenja biće uračunata *m* puta cena izlaznog konektora čvora *n*. Ovo je logično, s obzirom da se pravilo predstavljeno tim konektorom mora primeniti *m* puta da bi se došlo do rešenja.

Radi ilustacije prethodne diskusije posmatrajmo ponovo rešenje sa slike 27*a.* U cenu ovog rešenja, uračunata je dva puta cena pravila P5. Da bismo dobili rešenje, moramo cifru šest zameniti sa dve cifre 3, a zatim *svaku od* trojki zameniti primenom pravila P5. Prema tome, tokom zamene se dva puta primenjuje pravilo P5 pa je i logično da se cena ovog pravila uračuna dvostruko u cenu kompletnog rešenja.

Analogija A* algoritmu u slučaju pretrage korišćenjem AND-OR acikličkih grafova je algoritam pretrage AO*. Ovaj algoritam garantuje pronalaženje optimalnog rešenja kada se cena rešenja definiše na opisani način. Pretragu algoritmom AO* usmerava heuristička funkcija koja se definiše za svaki čvor AND/OR grafa. Heuristička funkcija za čvor *n* mora da ispunjava određeni uslov da bi pronađeno rešenje bilo optimalno, analogno slučaju kada se koristi algoritam A*. U slučaju algoritma AO* heuristička funkcija h(n) za proizvoljan čvor n AND/OR grafa mora da predstavlja potcenjenu cenu optimalnog podgrafa rešenja od čvora n do ciljnih čvorova.

Izvršavanje algoritma AO* sastoji se iz ponavljanja dve glavne faze:

- ekspanzije izabranog čvora grafa
- revizije funkcija procene čvorova grafa.

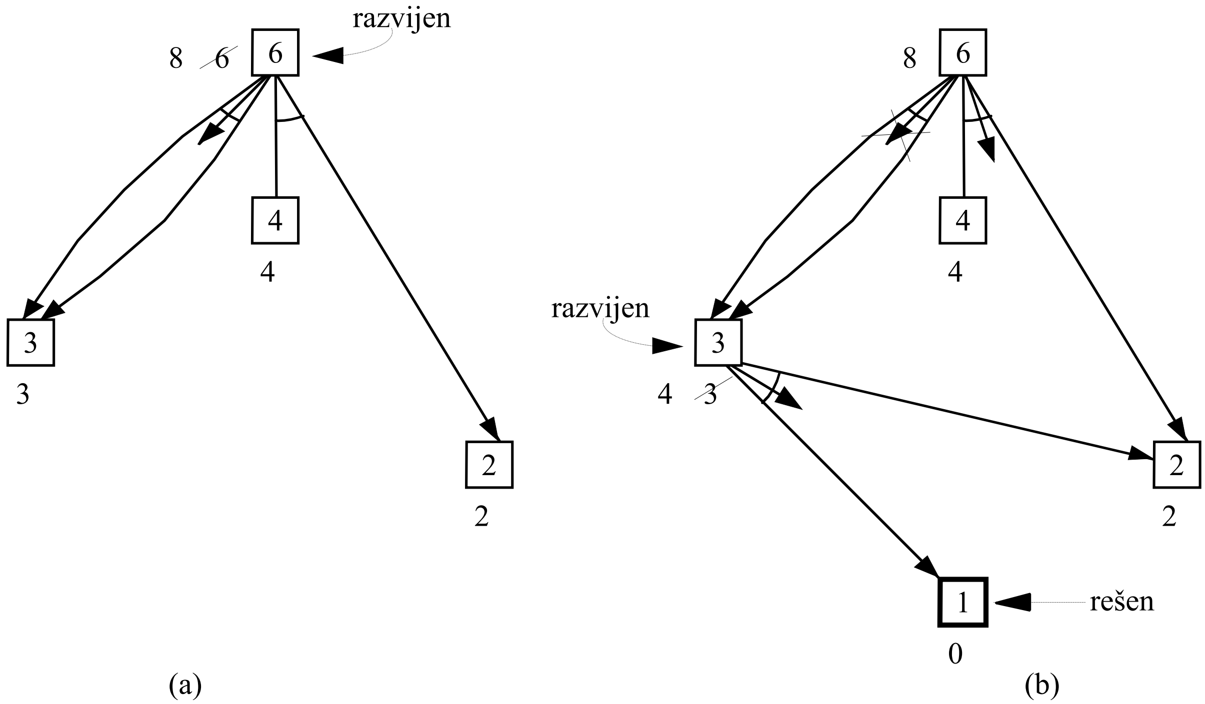
Prva faza je ekspanzija izabranog čvora AND/OR grafa i dodavanje njegovih izlaznih konektora i čvorova-naslednika u graf. Jedan od konektora, koji ima najbolju funkciju procene, pri tome biva obeležen kao rezultat druge glavne faze algorima. U svakom trenutku pretrage *podgraf najboljeg parcijalnog rešenja* (analogno najboljoj parcijalnoj putanji kod A* algoritma) može se dobiti polazeći od startnog čvora grafa i prateći obeležene konektore. Sledeći čvor koji će biti ekspandovan uvek je jedan od čvorova koji pripadaju podgrafu najboljeg parcijalnog rešenja koji nije ciljni čvor. Rešenje je nađeno kada se startni čvor obeleži kao REŠEN (čvor *n* je rešen kada su ekspandovani svi čvorovi koji nisu ciljni u podgrafu koji polazi od čvora *n* i ide preko obeleženih konektora do ciljnih čvorova).

Druga glavna faza algoritma je revizija funkcija procene čvorova u grafu. Funkcija procene f(*n*) čvora *n* predstavlja procenu cene podgrafa optimalnog rešenja od čvora *n* do skupa ciljnih čvorova. Kada se neki čvor *n* unese u graf, a pre nego što se taj čvor ekspanduje, njegova funkcija procene f(*n*) je inicijalno jednaka vrednosti njegove heurističke funkcije h(*n*). Za razliku od algoritma A*, gde se funkcija procene računa samo jedanput za svaki čvor stabla pretrage, u AO* algoritmu postoji potreba za revizijom funkcija procene čvorova. Kada se čvor *n* ekspanduje, njegova funkcija procene se ažurira na osnovu cena izlaznih konektora iz tog čvora i vrednosti funkcija procene čvorova naslednika. Ako iz čvora *n* ide više izlaznih konektora, uzima se najbolja (to jest najmanja) vrednost po nekom od konektora i taj konektor se obeležava čime se produžava najbolje parcijalno rešenje. Revidiranu vrednost funkcije procene čvora *n* potrebno je proslediti nagore u grafu. Pri tome se (eventualno) ažuriraju funkcije procene čvorova-prethodnika čvora *n* po *obleženim* konektorima. Čvorove-prethodnike po neobeleženim konektorima nije potrebno razmatrati. Pošto je procena uvek potcenjena veličina, a ažuriranjem se dobijaju preciznije procene, vrednost funkcije procene ažuriranjem može samo da se poveća. Ažuriranje može da dovede do toga da za određeni čvor funkcija procene po nekom od konektora koji nije obeležen postane povoljnija od vrednosti po obeleženom konektoru (s obzirom da se ova poslednja ažuriranjem povećala) , pa je tada potrebno premestiti obeležje na neobeleženi konektor.

Rešenje

Primenimo AO* algoritam na zadati problem. U postavci problema definisane su kako heuristička funkcija za svaki čvor AND/OR grafa, to jest za svaku cifru, tako i cene pojedinih konektora.

Graf pretrage se inicijalno sastoji samo od startnog čvora 6 za koga je vrednost funkcije procene jednaka vrednosti njegove heurističke funkcije i iznosi 6. Razvijanjem čvora 6 u graf unosimo izlazne konektore čvora 6 i čvorove 2, 3 i 4 (slika 28*a*).



Slika 28

Vrednosti funkcije procene za ove čvorove jednake su vrednostima njihovih heurističkih funkcija (na slici 28*a* trenutne vrednosti funkcija procene napisane su pored svakog čvora). Sada se revidira funkcija procene čvora 6. Po levom konektoru, vrednost funkcije procene za čvor 6 jednaka je dvostrukoj vrednosti funkcije procene čvora 3 (jer je to jedini čvor naslednik po levom 2-konektoru) na koju se dodaje cena levog konektora, odnosno:

$f_{P1} = c_{P1} + f(3) + f(3) = 2 + 3 + 3 = 8$

gde je sa P1 označen levi konektor na slici 28*a*. Na sličan način, vrednost funkcije procene po desnom konektoru je:

$f_{P2} = c_{P2} + f(4) + f(2) = 2 + 4 + 2 = 8$

Prema tome, ažurirana vrednost funkcije procene za čvor 6 je:

$f(6) = \min (f_{P1}, f_{P2}) = 8$

Potrebno je markirati jedan od konektora čvora 6 koji odgovara najboljem parcijalnom rešenju. S obzirom da su u konkretnom slučaju funkcije procene po oba konektora jednake proizvoljno je izabran levi konektor i obeležen strelicom. S obzirom da čvor 6 nema prethodnika u grafu, završena je faza ažuriranja funkcija procene čvorova.

U sledećoj iteraciji algoritma AO*, potrebno je za ekspanziju izabrati novi čvor. Taj čvor treba da se nalazi u podgrafu koji predstavlja najbolju (markiranu) parcijanu putanju. U konkretnom slučaju radi se o čvoru 3. Razvijanjem ovoga čvora nastaje situacija prikazana na slici 28*b*. U graf je unesen izlazni konektor čvora 3 koji odgovara pravilu P5. Od novih čvorova u graf je unesen čvor 1. Pošto je ovo ciljni čvor, on je odmah (šrafiranjem) označen kao REŠEN. Čvor 3 ne možemo još označiti kao rešen, jer nije rešen čvor 2. Nova vrednost funkcije procene za čvor 3 jednaka je:

$f(3) = c_{P5} + f(2) + f(1) = 2 + 2 + 0 = 4.$

Pošto je promenjena vrednost funkcije procene za čvor 3 potrebno je, u skladu sa algoritmom, ažurirati procenu za čvor 6 kao prethodnika čvora 3 po označenom markeru P1. Nova vrednost funkcije procene po konektoru P1 iznosi:

$f_{P1} = c_{P1} + f(3) + f(3) = 2 + 4 + 4 = 10$

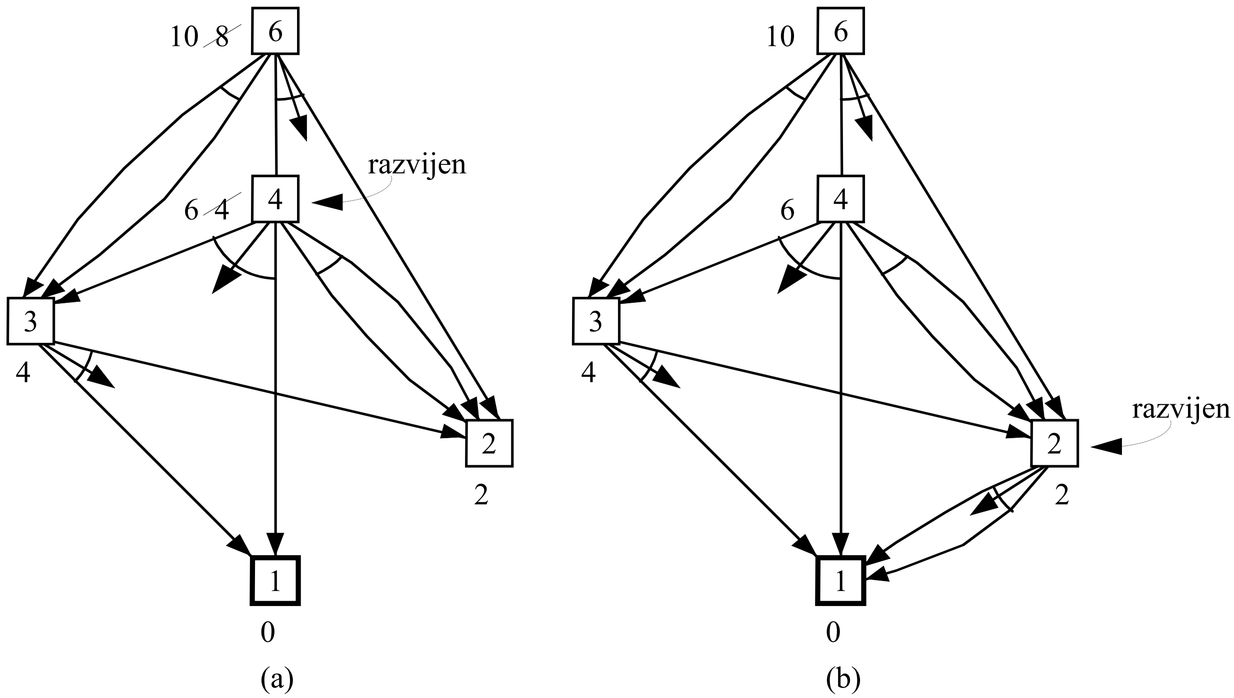
S obzirom da se funkcija procene po markeru P2 nije promenila i iznosi $f_{P2} = 8$, a pošto je

$f(6) = \min (f_{P1}, f_{P2}) = 8$

nema promene u vrednosti procene za čvor 6. Međutim, pošto se sada ova procena dobija po desnom konektoru, potrebno je ukloniti marker to jest strelicu sa konektora P1 i markirati konektor P2. Ovim je završena faza ažuriranja funkcija procene čvorova.

Pretraga se nastavlja razvijanjem jednog od nerazvijenih čvorova na markiranoj putanji. S obzirom da čvor 1 predstavlja ciljni čvor, on ne dolazi u obzir za razvijanje, tako da potrebno izabrati između čvorova 4 i 2.

Algoritam AO* ne specificira koji čvor treba izabrati pa je na proizvoljan način izabran čvor 4 (naravno, treba primetiti da je za dobijanje rešenja neophodno naknadno razviti i čvor 2 tako da ovaj izbor i nema preveliki uticaj na efikasnost pretrage). Slika 29*a* predstavlja graf pretrage posle razvoja čvora 4.



Slika 29

Funkcije procene za čvor 4 po konektorima P3 i P4 su:

$f_{P3} = c_{P3} + f(2) + f(2) = 2 + 2 + 2 = 6$

$f_{P4} = c_{P4} + f(3) + f(1) = 2 + 4 + 0 = 6$

pa je nova vrednost funkcije procene za čvor 4:

$f(4) = \min (f_{P3}, f_{P4}) = 6$

Na proizvoljan način (s obzirom na jednakost procena) između konektora P3 i P4 biramo P4 i markiramo ga. Sledi ažuriranje funkcije procene čvora 6, prethodnika čvora 4 po obeleženom konektoru. Nova vrednost f(6) je 10 jer su sada procene i po konektoru P1 i po konektoru P2 jednake 10. Konektor P2 ostaje obeležen jer nije dobijena bolja procena.

Sada se razvija poslednji od nerazvijenih čvorova a to je čvor 2 (slika 29*b*). Vrednost funkcije procene čvora 2 se ne menja jer je sada:

$f(2) = c_{P6} + f(1) + f(1) = 2 + 0 + 0 = 2$

pa nema potrebe ažurirati procene ostalih čvorova u grafu. Čvor 2 je rešen, jer je ciljni čvor 1 njegov jedini naslednik po izlaznom konektoru P6. Potrebno je razmotriti rešenost čvorova - prethodnika čvora 2 po obeleženim konektorima. Čvor 3 je rešen jer su oba njegova naslednika po obeleženom konektoru P5 rešena. Iz istih razloga rešeni su i čvorovi 4 i 6. Pošto je startni čvor 6 rešen, pretraga se završava. Funkcija procene startnog čvora daje tačnu cenu rešenja koja iznosi 10 u ovom slučaju. Dobijeno je rešenje sa slike 27*c*. S obzirom da smo u pojedinim trenucima pretrage birali konektore na proizvoljan način, mogli mogli smo dobiti i jedno od druga dva rešenja jer su jednako optimalna kao i dobijeno rešenje.