PLANIRANJE S PROLOGOM

INFORMATICA 4/91

Matjaž Likar in Nikola Guid
Tehniška fakulteta Maribor
Inštitut za računalništvo in informatiko
Smetanova 17, Maribor

Keywords: planning, search, heuristics

POVZETEK: Članek opisuje realizacijo sistemov za planiranje nedovisno od domene v programskem jeziku prolog. Za predstavitev problemov je uporabljena konceptualizacija v prostoru stanj. Planiranje poteka po treh različnih strategijah: planiranje z iskanjem v globino (sistem PLAN_S), hevristično planiranje z omejitvami (sistem PLAN_H) in planiranje z omejitvami (sistem PLAN_C). Rezultati eksperimentov kažejo na to, da je časovna zahtevnost sistema PLAN_S reda $O(b^d)$, kjer je b povprečni faktor vejitve in d dolžina plana. Sistema PLAN_H in PLAN_C uporabljata dodatene omejitve, ki izhajajo iz poznavanja posameznih domen. Rezultati eksperimentov nakazujejo, da je njuna časovna zahtevnost reda O(bd). Pri vseh treh sistemih je ugotovljena močna korelacija med časom planiranja in številom obravnavanih stanj. Za reševanje kompleksnejših problemov planiranja iz realnega sveta je potrebno uporabiti katero izmed boljših strategij.

ABSTRACT: Realization of the domain independent planning systems in PROLOG is described. State space conceptualization for problem representation is used. Three different planning strategies are implemented: planning with depth first search (system PLAN_S), heuristic planning with constraints (system PLAN_H) and planning with constraints (system PLAN_C). The results of the experimental measurements indicate that the time complexity of the system PLAN_S is $O(b^d)$, where b is an average branching factor and d plan length. System PLAN_H and PLAN_C use additional constraints coming out from better domain knowledge. Experimental measurements show that their time complexity is O(bd). Strong corelation between planning time and number of the visited states is established at all three systems. Better strategies should be used for solving planning problems with real life complexity.

1 UVOD

Začetki raziskav na področju planiranja v umetni inteligenci segajo v pozna petdeseta in zgodnja šestdeseta leta. Planiranje najprej pomeni način za poenostavitev zapletenih problemov, pri katerem iz rešitev enostavnih problemov razvijemo rešitve kompleksnih problemov. V tem času (1959) nastane eden izmed prvih sistemov za planiranja GPS ("general problem solver") [CHAP87,ALLE90]. V začetku sedemdesetih let se pojavijo nove definicije planiranja [BARR83,GEOR90], ki se ujemajo z običajnim razumevanjem te besede [RAND83,VERB87], in smatrajo plan kot natančen program akcij, planiranja pa kot razmišljanje o akcijah pred njihovim izvajanjem [NILS90]. Nadaljnji razvoj umetne inteligence prinese nove logične formalizme [GEOR90] (predikatni račun, situacijski račun, modalna logika), ki se izkažejo uporabni tudi na področju planiranja. Najbolj poznane sisteme za planiranje skupaj s približnimi časi njihovih nastankov prikazuje tabela 1 [TATE90]. Bistvene lastnosti, ki jih srečamo pri teh sistemih, so:

1. sposobnost izpeljevanja novih zaključkov iz že poznanih dejstev o problemu ter relacij in omejitev, ki veljajo za določeno domeno,
2. nelinearnost, s katero se izognemo iskanju vseh možnih vrstnih redov akcij znotraj plana,

3. hierarhičnost, ki proces planiranja razdeli na več nivojev, glede na stopnjo razgradnje problema,
4. sposobnost obravnavanja spremenljivk,
5. vključevanje dodatnih omejitev, s katerimi omejimo iskanje,
6. sposobnost ponovnega iskanja planov zaradi zunanjih sprememb, ki so nastale neodvisno od subjekta, ki izvršuje plan,
7. neodvisnost od domene.

Tabela 1 Kronologija nastankov poznanih sistemov planiranja

obdobje	ime sistema
1960-1965	GPS, QA3
1965-1970	REF-ARF, STRIPS
1970-1975	ABSTRIPS, HACKER, PLANEX, MACROPS, WARPLAN, HEARSAY, INTERPLAN
1975-1980	NOAH, NONLIN, MOLGEN, NONLIN+, NASL,
1980-1985	OPM, DEVISER, SIPE, PLANX 10, TWEAK
1985-1990	OPLAN, PRS, SCRAPS, CHEF, PLEX, GTD, FORBIN, PRIAR, PRODIGY/EBL

V tabeli 2 je podano, katere izmed zgoraj naštetih lastnosti so vključene v nekatere poznane sisteme za planiranje [WILK88].

Tabela 2 Lastnosti sistemov za planiranje

	lastnosti						
sistem	1	2	3	4	5	6	7
STRIPS							+
HACKER							+
ABSRIPST			+				+
NOAH		+	+	+			+
NONLIN		+	+	+			+
DEVISER		+	+	+	+		+
MOLGEN		+	+	+	+		
SIPE	+	+	+	+	+	+	+

Prvi sistemi za planiranje so predvsem eksperimentalne narave in imajo za cilj formalizacijo človekovih sposobnosti razmišljanja o prihodnjih akcijah. V sistemih STRIPS in NOAH je ideja planiranja realizirana kot iskanje potrebnih akcij in njihovo razvrščanje znotraj plana. Pri sistemu STRIPS, ki so ga razvijali ob koncu šestdesetih in v začetku sedemdesetih let, poteka to iskanje samo na enem nivoju, kar se pri kompleksnejših problemih izkaže kot premalo učinkovit pristop [WILK88]. Ta slabost je odpravljena v sistemu ABSTRIPS [SACE74], kjer poteka proces planiranja na večih nivojih **hierarhije abstraktnih** prostorov ("hiearchy of abstraction spaces"). Prvi klasični sistem za planiranje NOAH (1975) je že sposoben poiskati nelinearne in hierarhične plane, pri čemer uspešno obravnava tudi spremenljivke. Njegov naslednik NONLIN (1977) obvlada **vračanje** ("backtracking") [CHAP87,WILK88]. Začetki osemdesetih let prinesejo mnogi nove sisteme za planiranje: MOLGEN (1981) [STEF81a,STEF81b], DEVISER (1982) [CHAP87], SIPE (1983) [WILK88] in TWEAK (1984) [CHAP87]. Bistvo teh sistemov je, da upoštevajo dodatne **omejitve** ali **zahteve** ("constraints") nad spremenljivkami, s katerimi operiramo v procesu planiranja, omogočajo planiranje za doseg **konjunktivnih ciljev** ("conjunctive goals") in **ponovno planiranje** ("replanning") ob neuspešnem izvrševanju plana.

V tem članku želimo opisati realizacijo sistemov za planiranje in na podlagi eksperimentalno dobljenih podatkov ugotoviti, kakšna je časovna zahtevnost ter kateri faktorji vplivajo na učinkovitost planiranja. Poleg tega obravnavamo tudi lastne izkušnje in rešitve, ki so specifične za planiranje v prologu.

2 ZAHTEVNOST PLANIRANJA

Kompleksnost prostora stanj, ki ustreza problemu planiranja, je večinoma eksponentna funkcija velikosti problema [KORF87]. To potrjuje tudi kratka analiza velikosti prostora stanj za klasični problem planiranja v **svetu blokov** ("blocks world").

Število stanj, tj. različnih razporeditev n blokov v k skladov ($k \in \{1, 2, ..., n\}$) je enako vrednosti nepredznačenega Lahovega števila $L'_{n,k}$ (pri danem n in k) [AIGN79]:

$$L'_{n,k} = \frac{n!}{k!}\binom{n-1}{k-1} \qquad (1)$$

Celotno število različnih stanj za n blokov, tj. s(n), je enako vsoti vseh nepredznačenih Lahovih števil pri danem n (enačba 2, tabela 3):

$$s(n) = \sum_{k=1}^{n} L'_{n,k} \qquad (2)$$

Zgornja meja števila različnih stanj za svet blokov pa je:

$$s(n) = O\left((3/2)^{(n-1)}\, n!\right), \qquad (3)$$

iz česar vidimo, da gre za eksponentno rast prostora stanj. S povečevanjem števila blokov pa ne narašča le število različnih stanj, temveč tudi število vseh različnih akcij a(n). Če za premikanje blokov uporabljamo akcije: *unstack*, *stack* in *move* [GENE87], potem je:

$$a(n) = n \sum_{k=1}^{n-1} k\,(k+1)\, L'_{n-1,k} \qquad (4)$$

Povprečno število akcij, ki jih lahko izvedemo v nekem stanju, oz. **faktor vejitve** b(n) ("branching factor"), dobimo kot kvocient

števila vseh možnih akcij a(n) in števila vseh možnih stanj s(n) (tabela 3).

Tabela 3 Kompleksnost prostora stanj za svet blokov

n	s(n)	a(n)	b(n)
1	1	0	0.000
2	3	4	1.333
3	13	30	2.308
4	73	240	3.288
5	501	2 140	4.271
6	$4.051\ 10^{3}$	$2.130\ 10^{4}$	5.258
7	$3.763\ 10^{4}$	$2.351\ 10^{5}$	6.246
8	$3.944\ 10^{5}$	$2.854\ 10^{6}$	7.237
9	$4.597\ 10^{6}$	$3.782\ 10^{7}$	8.228
10	$5.894\ 10^{7}$	$5.434\ 10^{8}$	9.220
20	$3.277\ 10^{20}$	$6.283\ 10^{21}$	19.172
30	$1.980\ 10^{35}$	$5.771\ 10^{36}$	29.147
50	$3.772\ 10^{68}$	$1.853\ 10^{70}$	49.119
100	$2.422\ 10^{164}$	$2.400\ 10^{166}$	99.089

Pri večjem številu blokov se faktor vejitve približa vrednosti n, zato velja:

$$b(n) = O(n) \qquad (5)$$

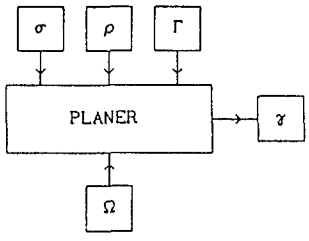
Zahtevnost problema planiranja in s tem tudi čas planiranja pa ni odvisen samo od števila blokov, oz. faktorja vejitve, marveč tudi od dolžine iskanega plana d, ki je lahko od 0 do 2(n-1). Funkcijski izraz, ki povezuje faktor vejitve in dolžino plana s časom planiranja, je odvisen od strategije planiranja.

3 REALIZACIJA SISTEMOV ZA PLANIRANJE

Študijo sistemov za planiranje smo izvedli v treh korakih. Najprej smo poiskali ustrezen model sistema, ga zatem implementirali na računalniku in preizkusili pri reševanju različno zahtevnih problemov.

3.1 Model sistema

Model sistema za planiranje (slika 1) smo zgradili na osnovi sheme iz [GENE87]. V sistem PLAN_S smo vgradili lastnosti 1, 4 in 7 (glej tabelo 2), kar pomeni, da gre za linearni sistem planiranja neodvisnega od domene z možnostjo obravnavanja spremenljivk. Sistem je v omejenem obsegu tudi sposoben izpeljevanja novih dejstev iz predhodno poznanih dejstev. Sistema PLAN_H in PLAN_C vsebujeta dodatne kontrolne mehanizme za povečevanje učinkovitosti (lastnost 5).

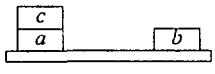


Slika 1 Zgradba sistema za planiranje

Oznake, ki nastopajo na sliki 1, pomenijo:

- σ - **označevalnik začetnega stanja** ("initial state designator"),
- ρ - **označevalnik cilja** ("goal designator") ali **ciljna relacija** ("goal relation"),
- Γ - množica **označevalnikov akcij** ("action designators") (enostavne akcije in zaporedja akcij),
- Ω - podatkovna baza s stavki o začetnem stanju, cilju in uporabnih akcijah,
- γ - plan (zaporedje akcij), s katerim ob izvršitvi v začetnem stanju dosežemo ciljno stanje.

Začetno stanje predstavlja posnetek opazovanega sveta v trenutku, preden začnemo izvajati akcije iz plana (slika 2).

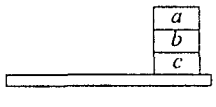


Slika 2 Primer začetnega stanja

Začetno stanje opišemo s seznamom dejstev, ki veljajo v tem stanju:

initial([clear(c),on(c,a),table(a),clear(b),table(b)]).

Cilj ("goal") je lahko katerokoli dosegljivo željeno stanje opazovanega sveta [GENE87] (slika 3). To je tisto stanje, ki ga v realnem svetu vzpostavimo tedaj, ko opravimo vse v planu zahtevane akcije. Definiciji cilja lahko ustreza več stanj.



Slika 3 Primer ciljnega stanja

Cilj, da je npr. v svetu treh blokov blok *a* na bloku *b* in hkrati blok *b* na bloku *c*, zapišemo kot:

goal([clear(a),on(a,b),on(b,c),table(c)]).

Za opisovanje akcij smo uporabili zapis, v katerem je podan seznam **pogojev** ("preconditions") za izvedbo akcije, seznam *pozitivnih* **učinkov** ("positive effects"), oz. dejstev, ki postanejo resnična po izvršitvi akcije, in seznam **negativnih učinkov** ("negative effects"), oz. dejstev, ki po izvršitvi akcije ne veljejo več.

action *action_name(argument1,argument2,...)* :
preconditions ==> *positive effects* # *negative effects.*

S tem formatom so implicitno rešeni **problemi ozadja** ("frame problems"), prav tako pa je v njem zajeta tudi strategija **poravnave stanj** ("state alignment") [GENE87,WINS84]. Tako smo se izognili spremenljivkam za označevanje stanj, ki so npr. potrebne pri planiranju z uporabo resolucije. Definicije akcij niso del sistema za planiranje, ampak predstavljajo njegove vhodne podatke (Γ na sliki 1). Podati jih moramo za vsako domeno posebej. Akcijo razlaganja (*unstack*) iz blokovnega sveta smo v naših sistemih definirali kot:

action u(X,Y) : [on(X,Y),clear(X)] ==>
[table(X),clear(Y)] # [on(X,Y)].

Sistem za planiranje iz vhodnih podatkov: označevalnika začetnega stanja σ, ciljne relacije ρ, množice označevalnikov akcij Γ in podatkovne baza Ω poišče plan γ, ki predstavlja rešitev problema planiranja tedaj in le tedaj, če zadovoljuje pogoja [GENE87]:

1. γ mora biti element množice označevalnikov akcij Γ:
$$\gamma \in \Gamma.$$
2. Podatkovna baze Ω mora logično implicirati, da izvršitev (Do) plana γ v začetnem stanju σ ustvari stanje, ki ustreza relaciji ρ:
$$\Omega \models \rho(Do(\gamma,\sigma)).$$

3.2 Implementacija sistemov

Sistem PLAN_S

Sistem PLAN_S smo realizirali na principu algoritma iskanja v globino v smeri naprej od začetnega stanja proti ciljnemu stanju (slika 4) [BRAT86,STER86]. Jedro tega sistema je rekurzivna procedura **plan(+Initial,+Goal,+History,+Depth,-Plan)** (s predpono + so označeni vhodni argumenti, z - pa izhodni argument). Pred klicem procedure moramo poznati začetno stanje, ciljno stanje in omejitev iskanja v globino. Argument **History** vsebuje seznam že poznanih stanj v delno zgrajenem planu in je pri prvem klicu prazen ([]). Iskanje v globino omejuje argument **Depth**. Robni pogoj rekurzije je izpolnjen tedaj, ko se začetno in ciljno stanje ujemata (**match**). Kot rezultat dobimo **Plan** - seznam akcij. Znotraj procedure **plan** kličemo proceduro za spremembo stanja **do_forw**, Z **do_forw(-Action,+OldState,-NewState)** preverimo, ali se pogoji **Pre** za izvedbo akcije **Action** ujemajo s starim stanjem **OldState**.

```
plan(S1,S2,_,_,[]) :- match(S1,S2).

plan(S1,S2,History,OldDepth,[Action|Plan]) :-
    OldDepth > 0,
    do_forw(Action,S1,NewS1),
    not(known(NewS1,History)),
    NewDepth is OldDepth - 1,
    plan(NewS1,S2,[S1|History],NewDepth,Plan).

do_forw(Action,OldState,NewState) :-
    action Action : Pre ==> Add # Del,
    match(Pre,OldState),
    del_all(Del,OldState,Temp),
    add_all(Add,Temp,NewState).
```

Slika 4 Planiranje z iskanjem v globino v smeri naprej

Zatem tvorimo novo stanje **NewState**, tako da iz starega stanja zbrišemo negativne učinke **Del** in mu dodamo pozitivne učinke **Add** akcije **Action**.

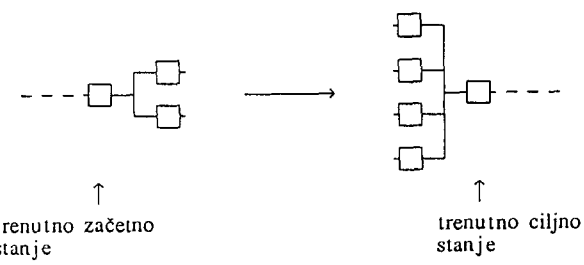
Sistem PLAN_H

Iskanje plana na slepo (s poskušanjem) in **vračanje** ("backtracking") v situacijah, kjer ugotovimo, da nadaljnje planiranje ni več mogoče, se je pri kompleksnejših primerih izkazalo kot premalo učinkovito. Sistem PLAN_H (slika 5) predstavlja nadgradnjo predhodnega sistema.

```
plan(_,S1,S2,[]) :- match(S1,S2).

plan(forward,S1,S2,[Action|Plan]) :-
    do_forw(Action,S1,NewS1,S2),
    direction(NewS1,S2,Direction),
    plan(Direction,NewS1,S2,Plan).

plan(backward,S1,S2,[Action|Plan]) :-
    do_back(Action,S2,NewS2,S1),
    direction(S1,NewS2,Direction)
    plan(Direction,S1,NewS2,P),
    concatenate(P,[Action],Plan).


do_forw(Action,OldState,NewState,Goal) :-
    choose(Action,OldState,NewState,Goal).

do_back(Inverse,OldGoal,NewGoal,Initial) :-
    choose(Action,OldGoal,NewGoal,Initial),
    inverse(Action,Inverse).

choose(Action,S1,NewS1,S2) :-
    findall(Act,(action Act : Pre ==> Add # Del,
                match(Pre,S1),legal(Act,S1,S2)),Actions),
    find_best(Actions,Action,S1,NewS1,S2).
```

Slika 5 Dvosmerno hevristično planiranje

S pomočjo procedure **direction(+S1,+S2,-direction)** smo realizirali dinamično izbiranje smeri planiranja. Na vsakem koraku iskanja plana ocenimo, ali terja manj naporov iskanje v smeri naprej ali v smeri nazaj. Če je faktor vejitve v trenutnem začetnem stanju manjši kakor faktor vejitve v trenutnem ciljnem stanju, potem naredimo en korak v smeri naprej (**do_forw**) (slika 6), v nasprotnem primeru pa v smeri nazaj (**do_back**).



Slika 6 Primer izbire smeri planiranja naprej

Sistemu PLAN_H smo dodali omejitev, s katero izmed vseh akcij, ki so izvedljive v nekem stanju, izberemo le dovoljene (**legal**) akcije. Akcija je v sistemu PLAN_H dovoljena v nekem stanju samo tedaj, ko njena izvršitev ne onemogoči nadaljnjega planiranja brez vračanja. Zaradi omejitve **legal** so učinki akcij **odvisni od okoliščin** ("context-dependent effects"), v katerih akcijo izvršujemo. Proceduri **legal** in **direction** je potrebno sestaviti za vsako domeno posebej. Pravimo, da gre za **kontrolne strategije iskanja za specifične domene** ("domain-specific search control strategies") [WILK88]. Iskanje plana v sistemu PLAN_H poteka po strategiji **vzpenjanje** ("hill-climbing"). Izmed vseh dovoljenih akcij v naslednjem koraku s hevristično ocenjevalno funkcijo izberemo najboljšo akcijo (**find_best** na sliki 5). Od hevristične ocenitvene funkcije zahtevamo, da nikoli ne **preceni** ("overestimates") dejanskih stroškov preostalega (še nedograjenega) plana. Izpolnitev te zahteve zagotavlja, da sistem PLAN_H vedno najde optimalno rešitev.

Sistem PLAN_C

Pri hevurističnem planiranju v sistemu PLAN_H je iskanje vseh potencialno možnih in glede na okoliščine dovoljenih akcij prostorsko in časovno zahteven proces, kajti učinkovita (hitra) izračunljivost hevristične ocenjevalne funkcije je slabo združljiva z njenim učinkovitim delovanjem. Sistem PLAN_C vsebuje samo omejitev **legal** in se pri izbiri naslednje akcije zadovolji s prvo najdeno dovoljeno akcijo (slika 7). S tem smo bistveno zmanjšali čas planiranja, odrekli pa smo se temu, da na vsakem koraku poiščemo najboljšo akcijo. Plani, ki jih najde sistem PLAN_C, zato niso optimalni.

```
choose(Action,S1,NewS1,S2) :-
    action Action : Pre ==> Add # Del,
    match(Pre,S1),
    legal(Action,S1,S2),
    del_all(Del,S1,Temp,add_all(Add,Temp,NewS1).
```

Slika 7 Izbira prve dovoljene akcije

V tabeli 4 je podan pregled omenjenih realiziranih sistemov s kratkim opisom strategije planiranja.

Tabela 4 Kratek opis realiziranih sistemov

sistem	opis sistema
PLAN_S	planiranje z iskanjem v globino naprej
PLAN_H	dvosmerno hevristično planir. z omejitvami
PLAN_C	dvosmerno planiranje z omejitvami

3.3 Eksperimenti

Vse eksperimente smo izvedli na računalniku PC s procesorjem 80386/80387 (25 MHz) in s pomnilnikom RAM velikosti 4 MB. Od sistemov smo zahtevali, da najdejo pravilen plan, poleg tega pa smo njihovo kvaliteto ocenjevali še po naslednjih treh kriterijih:

- čas planiranja,
- število obravnavanih stanj in
- dolžina najdenega plana.

Te podatke nam izpišejo sistemi v trenutku, ko je planiranje zaključeno. Za vse tri sisteme PLAN_S, PLAN_H in PLAN_C smo pripravili enak vzorec z 20 primeri iz sveta blokov, ki se razlikujejo po številu blokov ($n \in [1, 2, ..., 5]$) in dolžini plana d. V vzorcu so bili enakomerno zastopani primeri, ki so lažje rešljivi s planiranjem v smeri naprej, kot tudi primeri, ki so lažje rešljivi v smeri nazaj. Pazili smo tudi na to, da se v rešitvah - planih približno enako pogosto pojavljajo akcije *unstack*, *stack* in *move*. Pri sistemu PLAN_S smo za vsak problem planiranja najprej izbrali omejitev globine, ki je enaka dolžini optimalnega plana, zatem pa še omejitev 2(n-1), s katero lahko rešimo najbolj neugoden primer za n blokov. Sistema PLAN_H in PLAN_C smo dodatno preizkusili še na primerih z večjim številom blokov ($n \in [10, 20, 30, 50, 100]$), ker smo želeli ugotoviti, kje so meje njunih zmogljivosti.

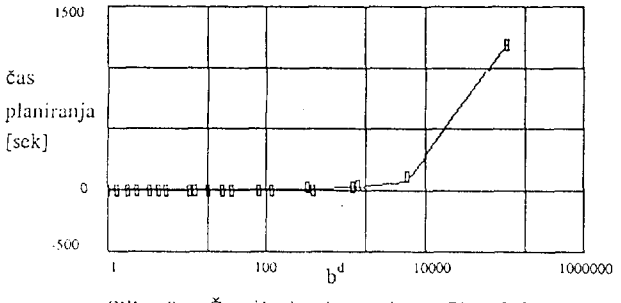
4 REZULTATI

V vzorcu 20-ih problemov planiranja je bilo več različnih problemov, vendar z enakim številom blokov (2, 3, 4 ali 5). Zanje smo izračunali srednjo vrednost dobljenih rezultatov in jih podali v tabelah 5, 6 in 7. Na dno tabel smo uvrstili tudi rezultate problemov planiranja za 10, 20, 30, 50 in 100 blokov. V primeru, da kakšen izmed sistemov ni našel neke rešitve zaradi prevelikih pomnilniških zahtev ali zaradi izjemno dolgega časa planiranja, smo to v tabelah označili s simbolom "-".

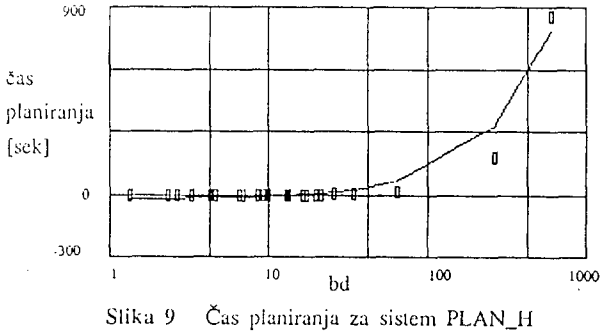
Tabela 5 Primerjava časa planiranja

	sistem			
	PLAN_S d = opt.	PLAN_S d = 2(n-1)	PLAN_H	PLAN_C
n	[min:sec]	[min:sec]	[min:sec]	[min:sec]
1	00:00.00	00:00.00	00:00.00	00:00.00
2	00:00.05	00:00.05	00:00.08	00:00.05
3	00:00.49	00:00.65	00:00.30	00:00.10
4	00:04.94	00:26.92	00:00.77	00:00.16
5	02:57.12	06:04.69	00:01.70	00:00.26
10	-	-	00:14.06	00:00.88
20	-	-	03:02.74	00:04.77
30	-	-	14:18.27	00:13.95
50	-	-	-	01:04.38
100	-	-	-	-

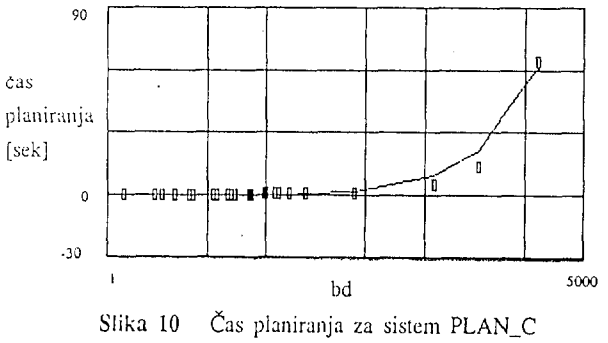
Slike 8, 9 in 10 prikazujejo izmerjene rezultate (pravokotniki) in predpostavljene časovne zahtevnosti (lomljenke), pri čemer je merilo na abscisi logaritemsko. Upoštevali smo vse rezultate eksperimentov in ne več njihovih srednjih vrednosti. Pri sistemu PLAN_S predstavlja diagram odvisnost časa planiranja od vrednosti b^d (slika 8). Odvisnosti časa planiranja od vrednosti bd za sistema PLAN_H in PLAN_C smo prikazali na slikah 9 in 10.



Slika 8 Čas planiranja za sistem PLAN_S



Slika 9 Čas planiranja za sistem PLAN_H



Slika 10 Čas planiranja za sistem PLAN_C

Koliko stanj so obravnavali (razvili ali obiskali) med planiranjem posamezni sistemi, prikazuje tabela 6.

Tabela 6 Primerjava števila obravnavanih stanj

	system			
	PLAN_S d = opt.	PLAN_S d = 2(n-1)	PLAN_H	PLAN_C
n	št. stanj	št. stanj	št. stanj	št. stanj
1	1	1	1	1
2	3.0	3.0	2.5	2.5
3	15.8	20.5	3.5	3.5
4	99.0	524.5	4.8	4.7
5	2181.3	4520.3	7.1	6.1
10	-	-	16	10
20	-	-	53	19
30	-	-	112	28
50	-	-	-	46
100	-	-	-	-

Pri sistemih za reševanje problemov planiranja je ena izmed važnih kvalitet, da sistem najde najkrajši plan. Dolžine najdenih planov za vse sistem smo podali v tabeli 7.

Tabela 7 Primerjava dolžine najdenega plana

	sistem			
	PLAN_S d = opt.	PLAN_S d = 2(n-1)	PLAN_H	PLAN_C
n	dol. plana	dol. plana	dol. plana	dol. plana
1	0	0	0	0
2	1.5	1.5	1.5	1.5
3	2.5	3.0	2.5	2.5
4	3.5	5.3	3.5	3.7
5	4.5	8.0	4.5	5.1
10	-	-	7	9
20	-	-	14	18
30	-	-	21	27
50	-	-	-	45
100	-	-	-	-

5 DISKUSIJA

V tem članku je podana primerjava delovanje treh različnih sistemov (strategij) za planiranje. Analiza učinkovitosti je opravljena na testnih primerih iz blokovnega sveta.

Rezultati eksperimentov se skladajo s teoretičnimi izhodišči, poleg tega pa nudijo tudi povsem praktične napotke, ki so uporabni pri snovanju sistemov za planiranje v prologu.

- Časovna zahtevnost planiranja z iskanjem v globino v sistemu PLAN_S se zelo dobro ujema (korelacijski faktor r = 0.999) s teoretično napovedanim redom $O(b^d)$ (slika 8).

- V nekaterih primerih je problem planiranja s sistemom PLAN_S lažje in hitreje rešljiv z večjo omejitvijo globine kot pa z manjšo. Ta anomalija se pojavi tedaj, ko že od vsega začetka razvijamo plan po pravi poti in ko do vračanja sploh ne pride.

- Učinkovitost sistema PLAN_S lahko povečamo z vgraditvijo dvosmernega iskanja, kot je to realizirano pri sistemih PLAN_H in PLAN_C.

- Rezultati eksperimentov s sistemoma PLAN_H in PLAN_C nakazujejo (r = 0.972 za PLAN_H in r = 0.973 za PLAN_C), da je časovna zahtevnost reda O(bd) (sliki 9 in 10), s čimer nam je omogočeno reševanje večjih problemov, kot s sistemom PLAN_S (tabela 5).

- Pri hevrističnem planiranju igra poleg sprejemljivosti [PEAR84] hevristične ocenitvene funkcije pomembno vlogo njena učinkovita izračunljivost. Za reševanje povprečnega problema planiranja s petimi bloki (tabeli 5 in 6) porabi sistem PLAN_H za obdelavo enega stanja (!) skoraj šestkrat več časa kot sistem PLAN_C in trikrat več časa kot sistem PLAN_S.

- Hevristično planiranje s sprejemljivo hevristično ocenitveno funkcijo in v povezavi z omejitvijo **legal** v sistemu PLAN_H nam da optimalen (najkrajši) plan.

- Obstaja tesna odvisnost časa planiranja od števila obravnavanih stanj. Statistična analiza to nakazuje s korelacijskimi faktorji r > 0.9 za vse tri sisteme.

Na področju planiranja v umetni inteligenci že obstajajo boljše strategije od teh, ki so opisane v tem članku. Omenimo samo nekatere izmed najperspektivnejših:

- razgradnja problemov v **podcilje** ("subgoals"): Rešitev celotnega problema dobimo s kompozicijo delnih rešitev.

- **makro operatorji** ('macro operators'): V problemih skušamo odkriti karakteristične vzorce - strukture, pri katerih smemo uporabiti določene makro operatorje.

- **hierarhično planiranje** ("hierarchical planning"): Probleme rešujemo na večih nivojih s pomočjo abstrakcije.

Za nadaljevanje razvoja sistemov planiranja neodvisnega od domene menimo, da bo potrebno uporabiti kombinacijo večih strategij. Smatramo, da bi bila ena izmed perspektivnih možnosti hierarhični pristop k reševanju problemov, kjer bi na vsakem nivoju lahko uporabljali vnaprej definirane enostavne ali makro operatorje, sama izbira operatorjev pa bi lahko potekala z uporabo učinkovitih hevrističnih ocenitvenih funkcij. Sistemi bi morali biti zgrajeni tako, da bi 'razumeli' primerno formalizirano znanje iz različnih področij, hkrati pa bi morali znati to znanje tudi učinkovito uporabljati.

LITERATURA

[AIGN79] Aigner, M., *Combinatorial Theory*. New York: Springer-Verlag, 1979.

[ALLE90] Allen, J., Hendler, J., and Tate, A. (eds.), *Readings in Planning*. San Mateo, California: Morgan Kaufman, 1990.

[BARR83] Barr, A. and Feigenbaum, E. A., *The Handbook of Artificial Intelligence*, Vol. III. Los Altos, California: William Kaufmann, 1983.

[BRAT86] Bratko, I., *Prolog Programming for Artificial Intelligence*. Wokingham, England: Addison-Wesley, 1986.

[CHAP87] Chapman, D., "Planning for Conjunctive Goals", *Artificial Intelligence*, 32(3): 333- 337, 1987.

[GENE87] Genesereth, M. R. and Nilsson, N. J., *Logical Foundations of Artificial Intelligence*. Los Altos, California: Morgan Kaufman, 1987.

[GEOR90] Georgeff, M. P., "Planning", in Allen, J., Hendler, J., and Tate, A. (eds.), *Readings in Planning*. San Mateo, California: Morgan Kaufman, 1990, pp. 5-25.

[KORF87] Korf, R. E., "Planning as Search: A Quantitative Approach", *Artificial Intelligence*, 33(1): 65-88, 1987.

[NILS90] Nilsson, N. J., "Foreword", in Allen, J., Hendler, J., and Tate, A. (eds.), *Readings in Planning*. San Mateo, California: Morgan Kaufman, 1990, pp. xi-xii.

[PEAR84] Pearl, J., *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Reading, Massachusetts: Addison-Wesley, 1984.

[RAND83] *The Random House Dictionary of the English language*. New York: Random House, 1983.

[SACE74] Sacerdoti, D. E., "Planning in a Hierarchy of Abstraction Spaces", *Artificial Intelligence*, 5(2): 115-135, 1974.

[STEF81a] Stefik, M., "Planning with Constraints (MOLGEN: Part 1)", *Artificial Intelligence*, 16(2): 111-140, 1981.

[STEF81b] Stefik, M., "Planning and Meta-Planning (MOLGEN: Part 2)", *Artificial Intelligence*, 16(2): 141-170, 1981.

[STER86] Sterling, L. and Shapiro, E., *The Art of Prolog: Advanced Programming Techniques*. Cambridge, Massachusetts: MIT Press, 1986.

[TATE90] Tate, A., "A Review of AI Planning techniques", in Allen, J., Hendler, J., and Tate, A. (eds.), *Readings in Planning*. San Mateo, California: Morgan Kaufman, 1990, pp. 26-49.

[VERB87] Verbinc, F., *Slovar tujk*. Ljubljana: Cankarjeva založba, 1987.

[WILK88] Wilkinson, D. E., *Practical Planning: Extending the Classical AI Planning Paradigm*. San Mateo, California: Morgan Kaufman, 1988.

[WINS84] Winston, P. H., *Artificial Intelligence*. Reading, Massachusetts: Addison-Wesley, 1984.