Pretrage s nedeterminističkim akcijama

Anita Jukić, Katica Babić, Manuela Pavić

ODJEL ZA MATEMATIKU
SVEUČILIŠTE J. J. STROSSMAYERA U OSIJEKU

11. lipnja 2012.

Sadržaj

Sadržaj

Sadržaj

Sadržaj

Sadržaj

Sadržaj

- Okoliš je potpuno vidljiv i determiniziran (predodređen) i agent (može i agens) zna koji su učinci svakog postupka (akcije).
- Kada je okoliš djelomično vidljiv ili neodređen (nedeterminiziran) ili oboje, predmeti opažanja postaju korisni.
- U djelomično vidljivom okolišu svaki predmet opažanja pomaže suziti broj mogućih stanja u kojima bi agent mogao biti čineći tako da agentu bude lakše postići njegove ciljeve. Kada je okoliš nedeterminiziran, predmeti opažanja govore agentu koji se od mogućih ishoda njegovih postupaka zapravo pojavio.

Nestalan svijet usisivača

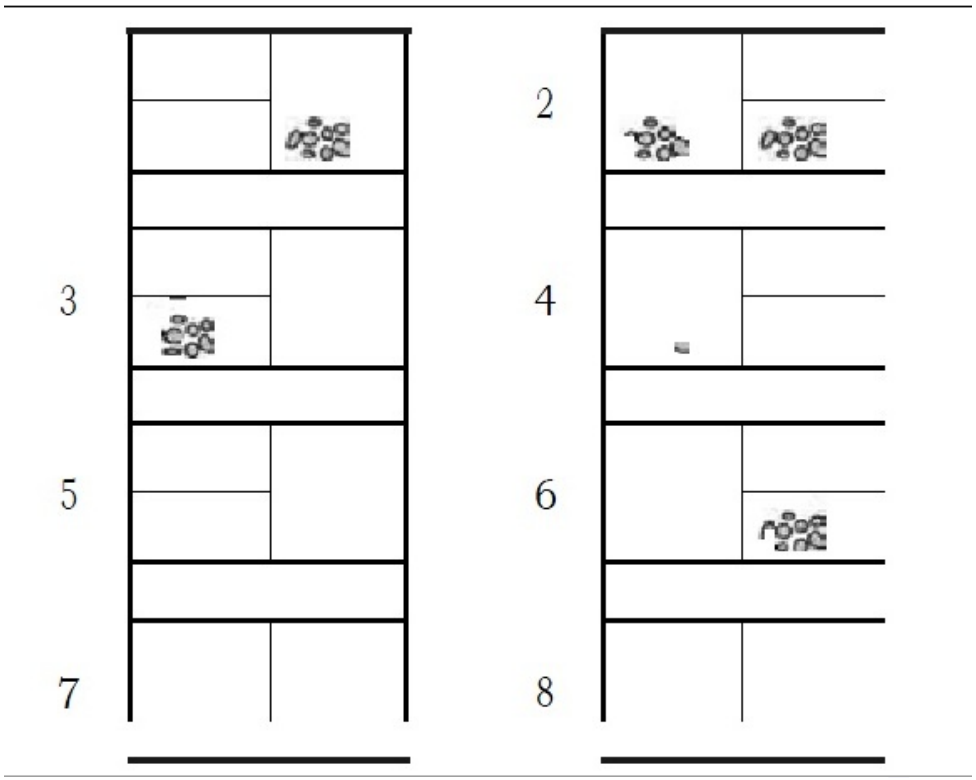
Nestalan svijet usisivača

Kao primjer upotrijebit ćemo svijet usisivača koji ćemo definirati kao problem pretraživanja.
To se može oblikovati kao sljedeći problem:

- stanja
- početno stanje
  - akcije
  - prijelazni model
  - test cilja
  - vrrijednost puta

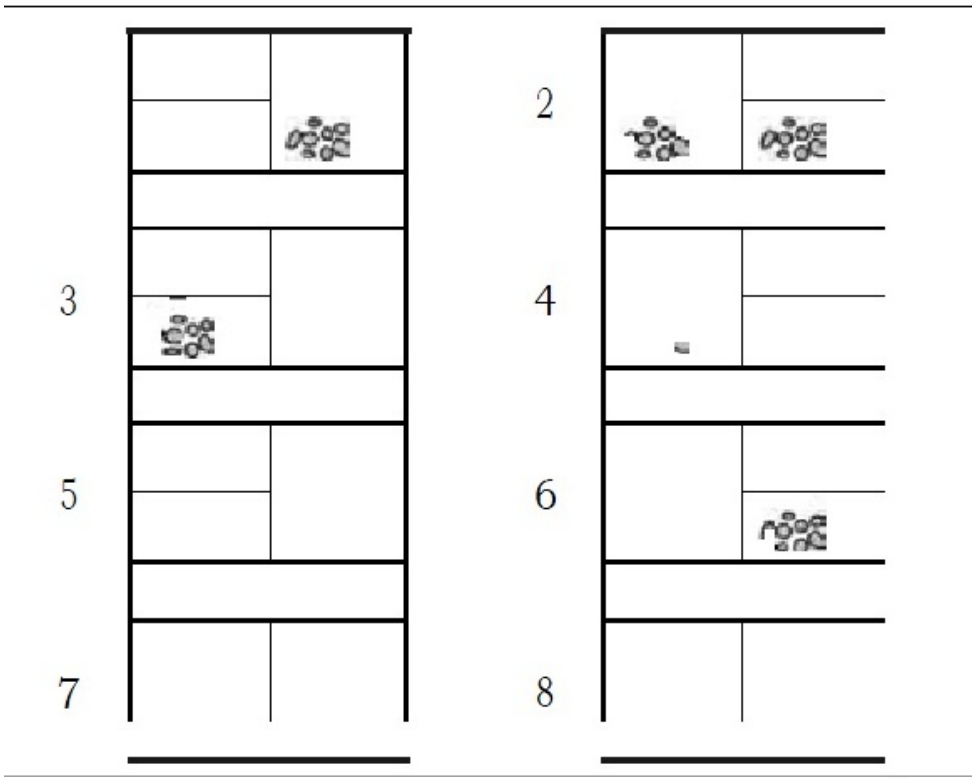
Nestalan svijet usisivača

Prisjetimo se da *prostor stanja* ima osam stanja:



Nestalan svijet usisivača

Prisjetimo se da *prostor stanja* ima osam stanja:

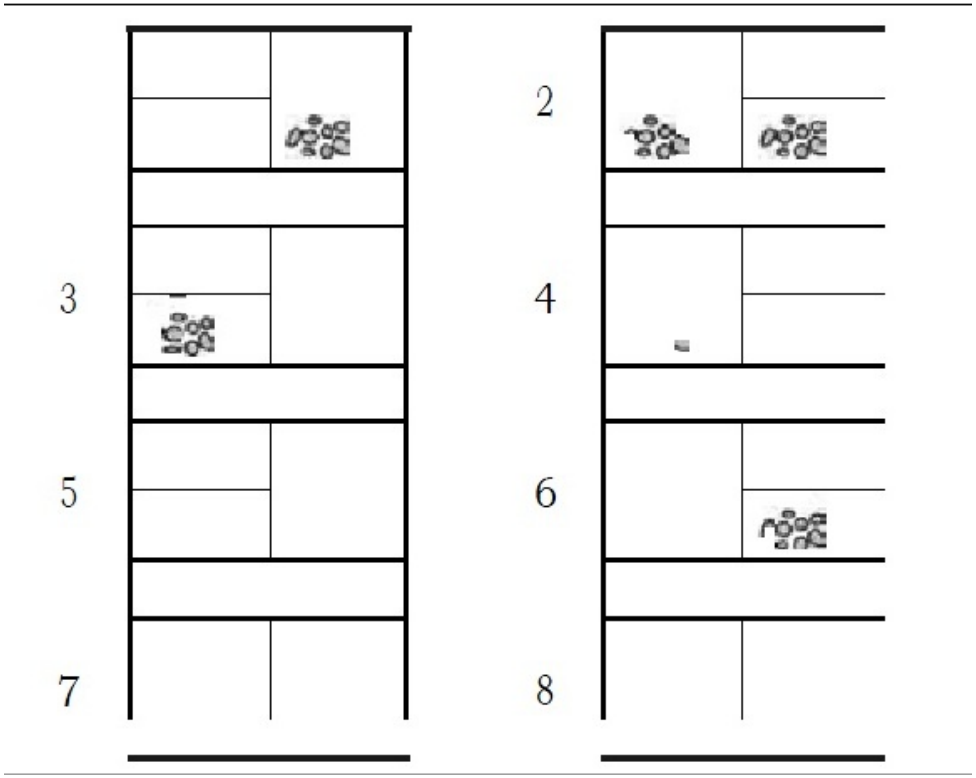


Postoje tri akcije: *lijevo, desno i usiši*, a cilj je počistiti svu prljavštinu (nečistoću). Ako je okoliš vidljiv, determiniziran i potpuno poznat onda je problem lako rješiv pomoću bilo kojeg algoritma pretraživanja i rješenje je postupni niz (postupak redoslijeda).
Sada pretpostavimo da upoznajemo nedeterminizam u obliku snažnog, ali nestalnog usisavača.

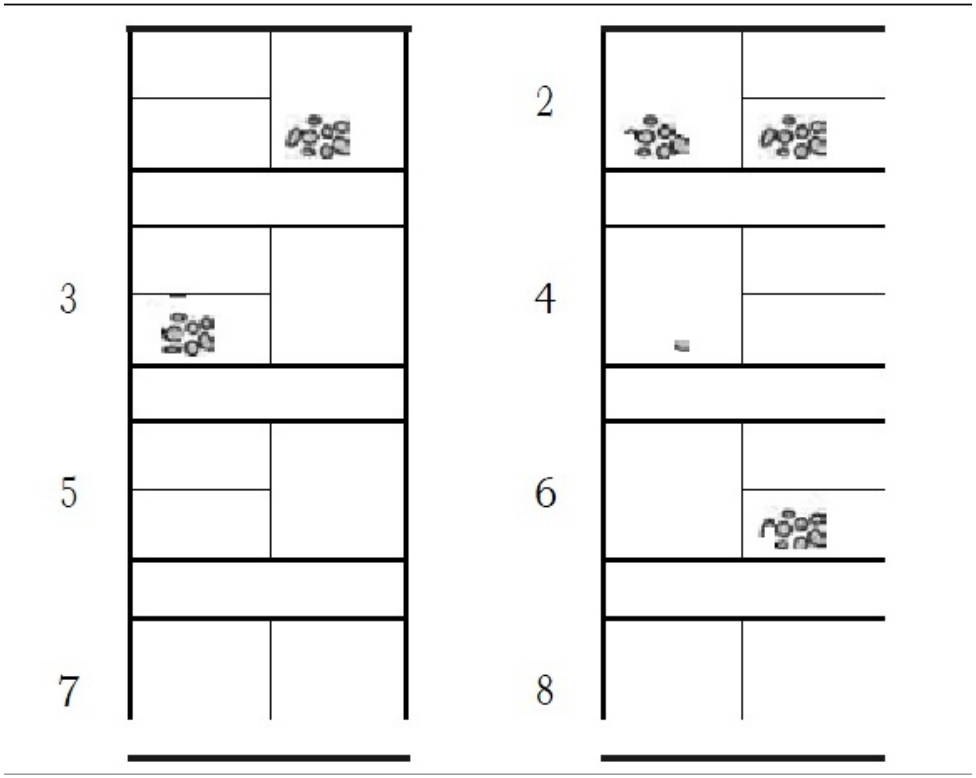
U nestalnom svijetu usisivača akcija usiši radi na sljedeći način:

1. kada se primijeni na onečišćeni kvadratić akcija čisti kvadratić i ponekad čisti prljavštinu u susjednom kvadratiću također
2. kada se akcija rabi na čistom kvadratiću ponekad ostavlja nečistoću na tepihu

Nestalan svijet usisivača

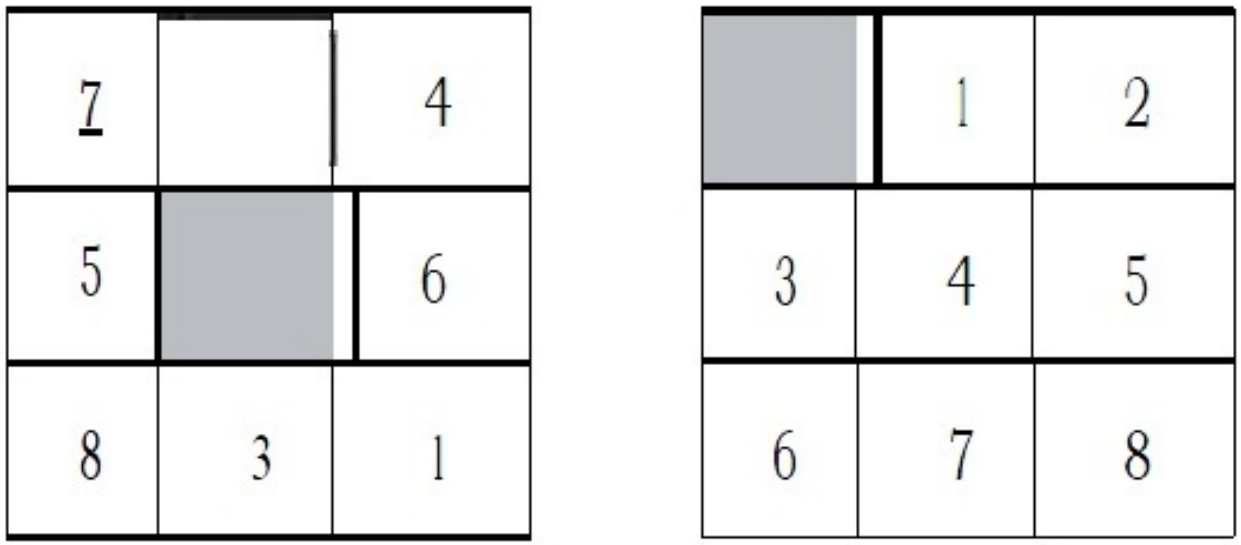


Nestalan svijet usisivača



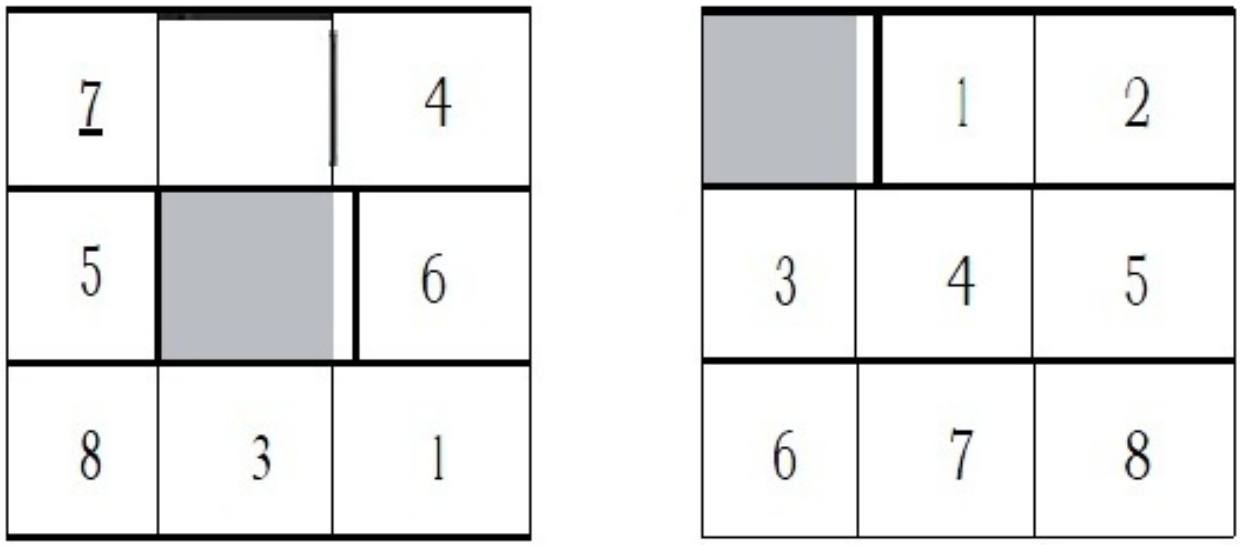
Osmodijelne puzle

Instanca koja je prikazana na slici 2. sastoji se od 3x3 ploče s osam stupaca i praznih područja.



Osmodijelne puzle

Instanca koja je prikazana na slici 2. sastoji se od 3x3 ploče s osam stupaca i praznih područja.



Stupac susjedan praznom području može otklizati u prostor. Predmet je dosegnuti određeni cilj. Formulacija je ovakva:

- stanja
- početno stanje
  - akcije
  - prijelazni model
  - test cilja
  - vrrijednost puta

Koja smo izlučivanja uključili ovdje? Akcije su određene svojim početnim i konačnim stanjima ignorirajući središnje lokacije gdje zastoj kliže. Imamo i akcije kao što su prodrmavanje hrpe kada dijelovi zapnu i izvlačenje dijelova nožem i vraćanje natrag. Ispustili smo opis pravila puzle izbjegavajući sve detalje fizičke manipulacije.

I-ILI stabla pretraživanja

I-ILI stabla pretraživanja

Kako pronaći rješenja za nedeterminizirani problem?

I-ILI stabla pretraživanja

Kako pronaći rješenja za nedeterminizirani problem?

- stablo pretraživanja

I-ILI stabla pretraživanja

Kako pronaći rješenja za nedeterminizirani problem?

- stablo pretraživanja
- ILI-čvorovi

I-ILI stabla pretraživanja

Kako pronaći rješenja za nedeterminizirani problem?

- stablo pretraživanja
- ILI-čvorovi
- I-čvorovi

I-ILI stabla pretraživanja

Kako pronaći rješenja za nedeterminizirani problem?

- stablo pretraživanja
- ILI-čvorovi
- I-čvorovi

I-ILI stabla pretraživanja

Rješenje za I-ILI problem je podstablo koje:

Rješenje za I-ILI problem je podstablo koje:

- ima ciljne čvorove na svakom listu

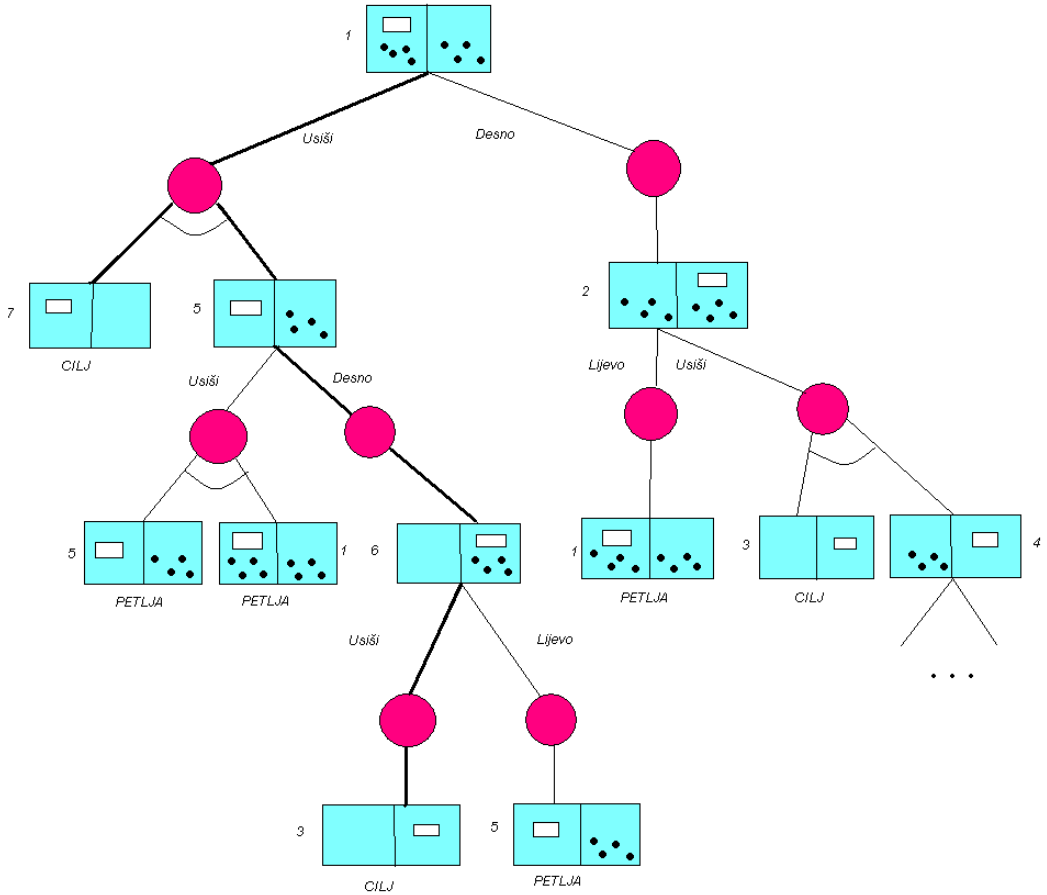
Rješenje za I-ILI problem je podstablo koje:

- ima ciljne čvorove na svakom listu
- određuje jednu akciju na svakom od ILI-čvorova

Rješenje za I-ILI problem je podstablo koje:

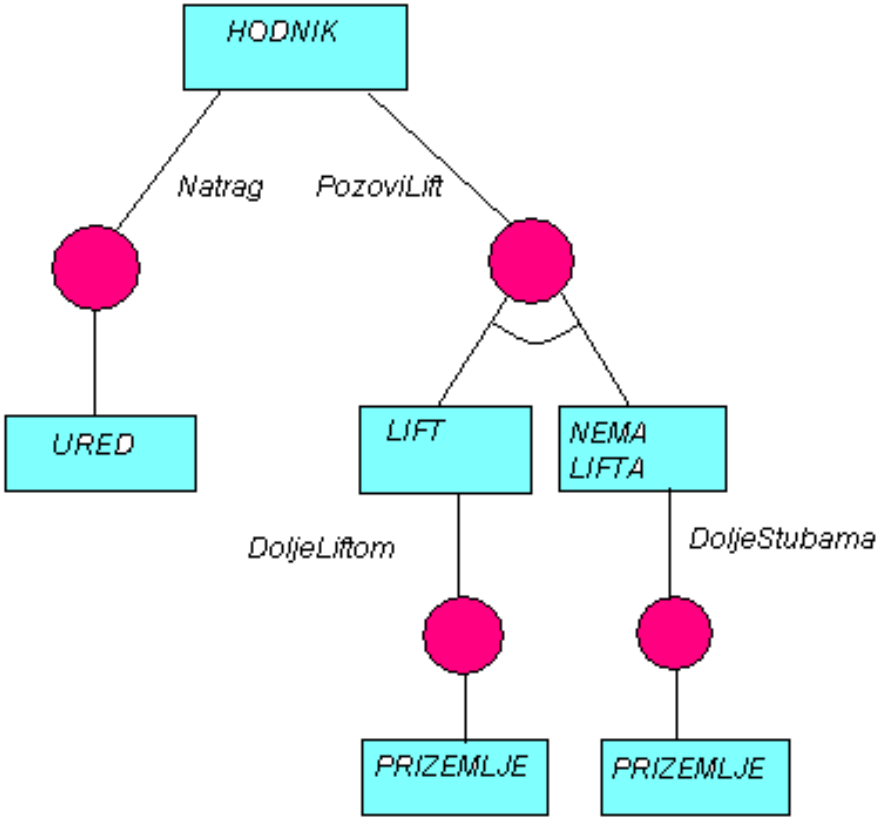
- ima ciljne čvorove na svakom listu
- određuje jednu akciju na svakom od ILI-čvorova
- uključuje svaku granu ishoda na kraju svakog I-čvora

1. PRIMJER



Rješenje je prikazano podebljanim linijama.

2. PRIMJER:


HODNIK
Natrag
PozoviLift
URED
LIFT
NEMA LIFTA
DoljeLiftom
DoljeStubama
PRIZEMLJE
PRIZEMLJE

Algoritam

Algoritam

- algoritam za traženje I-ILI grafova (stabala) u nedeterminističkoj okolini

Algoritam

- algoritam za traženje I-ILI grafova (stabala) u nedeterminističkoj okolini
- vraća uvjetni plan koji dolazi do željenog cilja u svim okolnostima

Algoritam

- algoritam za traženje I-ILI grafova (stabala) u nedeterminističkoj okolini
- vraća uvjetni plan koji dolazi do željenog cilja u svim okolnostima
- sastoji se od tri funkcije: glavne funkcije AND-OR-GRAPH-SEARCH i pomoćnih funkcija OR-SEARCH i AND-SEARCH (koje zapravo rješavaju problem, a pozivaju se unutar glavne funkcije)

Algoritam

Algoritam

Ovo je glavna funkcija:
function AND-OR-GRAPH-SEARCH (*problem*) **returns** a conditional plan, or failure
OR-SEARCH(*problem*.INITIAL-STATE,*problem*,[])

Algoritam

Algoritam

Pomoćna funkcija:
function OR-SEARCH(*state*, *problem*, *path*) **returns** a conditional plan, or failure
if *problem*.GOAL-TEST(*state*) **then return** the empty plan
if *state* is on *path* **then return** failure
for each *action* **in** *problem*.ACTIONS(*state*) **do**
plan $\leftarrow$ AND-SEARCH(RESULTS(*state*, *action*), *problem*,[*state* | *path*])
if *plan* $\neq$ *failure* **then return** [*action* | *plan*]
return *failure*

Algoritam

Algoritam

Pomoćna funkcija:
function AND-SEARCH(*states*, *problem*, *path*) **returns** a conditional plan, or failure
for each s_i **in** *states* **do**
$plan_i \leftarrow$ OR-SEARCH(s_i,*problem*, *path*)
if *plan* = *failure* **then return** *failure*
return [if s_1 then $plan_1$ else if s_2 then $plan_2$ else . . . if s_{n-1} then $plan_{n-1}$ else $plan_n$]

Pronalaženje cilja

Pronalaženje cilja

AND-OR grafovi u nedeterminističkoj okolini mogu biti istraživani pomoću pretraživanja u dubinu i širinu.

Pronalaženje cilja

AND-OR grafovi u nedeterminističkoj okolini mogu biti istraživani pomoću pretraživanja u dubinu i širinu.

- **Pretraživanje u dubinu** (engl. depth-first search, DFS) je slijepa strategija istraživanja koja ne obilazi čvorove po razinama, nego najprije obilazi sve sljedbenike nekog čvora, a tek nakon što se dođe do dna grafa, pretraživanje se usmjerava na sljedeći čvor na istoj razini.

Pronalaženje cilja

AND-OR grafovi u nedeterminističkoj okolini mogu biti istraživani pomoću pretraživanja u dubinu i širinu.

- **Pretraživanje u dubinu** (engl. depth-first search, DFS) je slijepa strategija istraživanja koja ne obilazi čvorove po razinama, nego najprije obilazi sve sljedbenike nekog čvora, a tek nakon što se dođe do dna grafa, pretraživanje se usmjerava na sljedeći čvor na istoj razini.

Algoritam za pretraživanje AND-OR grafova u nedeterminističkoj okolini daje **rekurzivan** *depth – first* algoritam za AND-OR graf.

function OR-SEARCH(*state*, *problem*, *path*) **returns** a conditional plan, or failure

  if *problem*.GOAL-TEST(*state*) **then return** the empty plan
  if *state* is on *path* **then return** failure
  for each *action* **in** *problem*.ACTIONS(*state*) **do**
    plan $\leftarrow$ AND-SEARCH(RESULTS(*state*, *action*), *problem*,[*state* | *path*])
    if *plan* $\neq$ *failure* **then return** [*action* | *plan*]
  return *failure*

---

function depthFirstSearch(*s*, *succ*, *goal*)

  if *goal(s)* **then return** s
  for $m \in succ(s)$ **do**
    $r \leftarrow$ depthFirstSearch(*m*, *succ*, *goal*)
    if $r \neq$ *fail* **then return** *r*
  return *fail*

Jedan aspekt algoritma je način na koji se on nosi s ciklusima koji se često javljaju u nedeterminiziranim problemima.
Ako je trenutno stanje identično kao na putu od korijena onda se vraća s neuspjehom, a ako je početno stanje identično stanju na putu od korijena, onda vraća pogrešku.

Jedan aspekt algoritma je način na koji se on nosi s ciklusima koji se često javljaju u nedeterminiziranim problemima.
Ako je trenutno stanje identično kao na putu od korijena onda se vraća s neuspjehom, a ako je početno stanje identično stanju na putu od korijena, onda vraća pogrešku.

Ovom provjerom osiguravamo da algoritam završava u svakom konačnom stanju jer svaki put mora doseći cilj, slijepu ulicu ili stanje koje se ponavlja.

Jedan aspekt algoritma je način na koji se on nosi s ciklusima koji se često javljaju u nedeterminiziranim problemima.
Ako je trenutno stanje identično kao na putu od korijena onda se vraća s neuspjehom, a ako je početno stanje identično stanju na putu od korijena, onda vraća pogrešku.

Ovom provjerom osiguravamo da algoritam završava u svakom konačnom stanju jer svaki put mora doseći cilj, slijepu ulicu ili stanje koje se ponavlja.

Uočimo da algoritam ne provjerava je li trenutno stanje ponavljanje nekog drugog stanja na putu od korijena što je važno za učinkovitost.

Pretraživanje u širinu

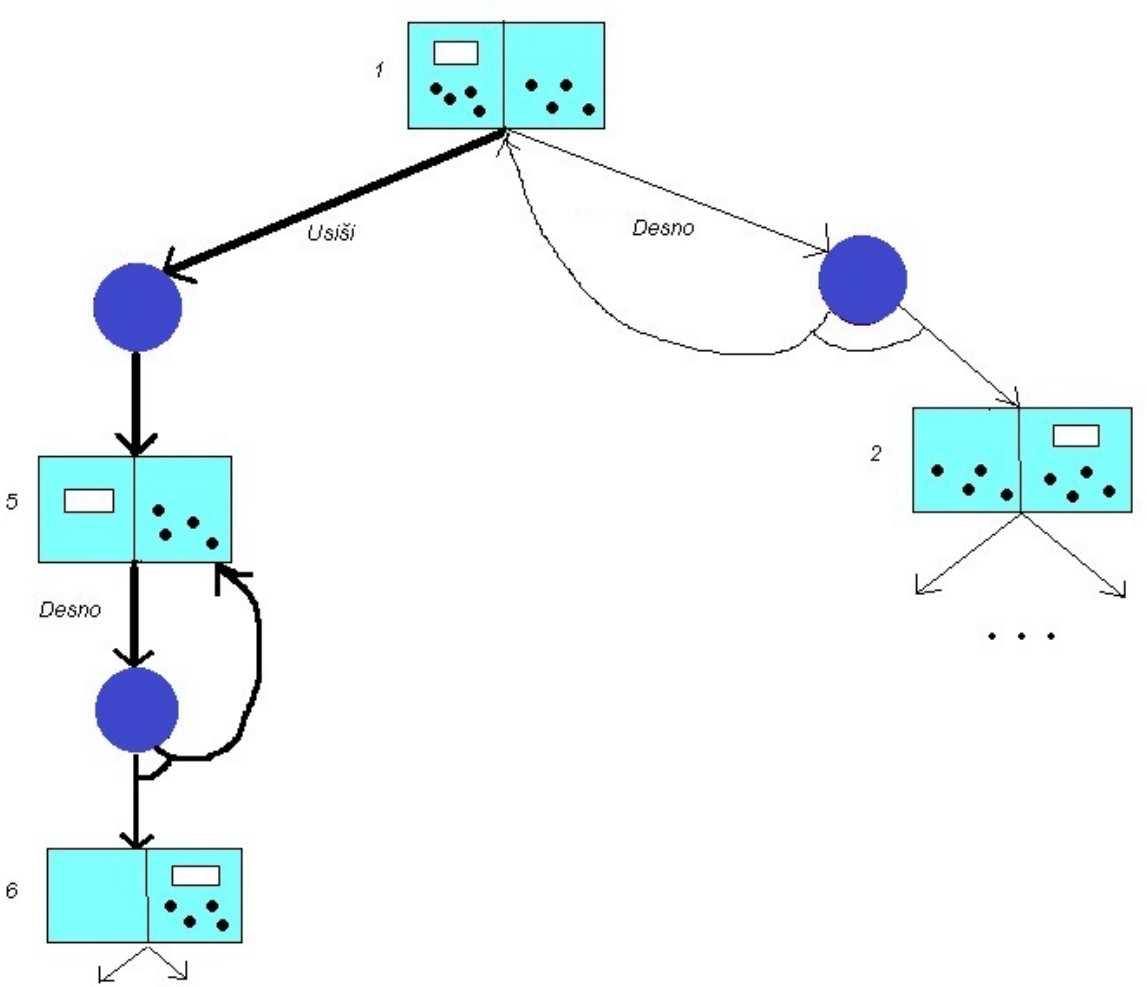
Pretraživanje u širinu

- **Pretraživanje u širinu** (engl. breadth-first search, BFS) je jednostavna slijepa strategija pretraživanja u kojoj se nakon ispitivanja korijenskog čvora, obilaze sva njegova djeca, a potom i sva njegova djeca itd.

Pretraživanje u širinu

- **Pretraživanje u širinu** (engl. breadth-first search, BFS) je jednostavna slijepa strategija pretraživanja u kojoj se nakon ispitivanja korijenskog čvora, obilaze sva njegova djeca, a potom i sva njegova djeca itd.
  Ovdje će se prvo ispitati svi čvorovi jedne razine, a tek će se nakon toga, prijeći na ispitivanje čvorova na sljedećoj razini.

- Slika prikazuje dio grafa pretraživanja za nesiguran svijet vakuuma gdje su eksplicitno prikazani neki ciklusi.



Razmatrajmo svijet vakuuma koji je identičan uobičajenom svijetu vakuuma osim toga što akcije pokreta ponekad dožive neuspjeh, ostavljajući agenta na istom mjestu.
Npr, micanje desno u stanju 1 vodi do skupa stanja (1, 2). Na prehodnoj slici mogli smo vidjeti da iz stanja 1 više nema necikličkih rješenja i AND-OR graf pretraživanja vraća grešku.

Razmatrajmo svijet vakuuma koji je identičan uobičajenom svijetu vakuuma osim toga što akcije pokreta ponekad dožive neuspjeh, ostavljajući agenta na istom mjestu.
Npr, micanje desno u stanju 1 vodi do skupa stanja (1, 2). Na prehodnoj slici mogli smo vidjeti da iz stanja 1 više nema necikličkih rješenja i AND-OR graf pretraživanja vraća grešku. Međutim, postoji cikličko rješenje koje ima svrhu pokušavati stalno desno dok god funkcionira.
Ono glasi:

$$[\textit{Usiši}, \textit{LI}: \textit{Desno}, \textbf{if } \textit{Stanje} = 5 \textbf{ then } L_1 \textbf{ else } \textit{Usiši}]$$

Bolja sintaksa za "petljajući" dio ovog plana bila bi:

$$[\textbf{while } \textit{Stanje=5} \textbf{ do } \textit{Desno}].$$

- Općenito, ciklički plan mogao bi se smatrati rješenjem ako je svaki list ciljno stanje i ako je list dostupan iz svakog dijela plana.

- Općenito, ciklički plan mogao bi se smatrati rješenjem ako je svaki list ciljno stanje i ako je list dostupan iz svakog dijela plana.
- Ključno shvaćanje je da se petlja iz stanja prostora u stanje L translatira u planiranu petlju sve do točke gdje se izvodi podplan za stanje L.

- Općenito, ciklički plan mogao bi se smatrati rješenjem ako je svaki list ciljno stanje i ako je list dostupan iz svakog dijela plana.
- Ključno shvaćanje je da se petlja iz stanja prostora u stanje L translatira u planiranu petlju sve do točke gdje se izvodi podplan za stanje L.
- Agent koji izvodi cikličko rješenje s vremenom će doseći cilj ako se svaki ishod nedeterminističke akcije s vremenom pojavi.

- Općenito, ciklički plan mogao bi se smatrati rješenjem ako je svaki list ciljno stanje i ako je list dostupan iz svakog dijela plana.
- Ključno shvaćanje je da se petlja iz stanja prostora u stanje L translatira u planiranu petlju sve do točke gdje se izvodi podplan za stanje L.
- Agent koji izvodi cikličko rješenje s vremenom će doseći cilj ako se svaki ishod nedeterminističke akcije s vremenom pojavi.
- Je li ovaj uvjet razuman?

- Općenito, ciklički plan mogao bi se smatrati rješenjem ako je svaki list ciljno stanje i ako je list dostupan iz svakog dijela plana.
- Ključno shvaćanje je da se petlja iz stanja prostora u stanje L translatira u planiranu petlju sve do točke gdje se izvodi podplan za stanje L.
- Agent koji izvodi cikličko rješenje s vremenom će doseći cilj ako se svaki ishod nedeterminističke akcije s vremenom pojavi.
- Je li ovaj uvjet razuman?
  Ovisi o razlogu za nedeterminizam.

Zaključak

- Okoliš može biti potpuno vidljiv i determiniziran i tada agent zna koji su učinci svake akcije, ili može biti djelomično vidljiv ili nedeterminiziran ili oboje, predmeti opažanja postaju korisni.

Zaključak

- Okoliš može biti potpuno vidljiv i determiniziran i tada agent zna koji su učinci svake akcije, ili može biti djelomično vidljiv ili nedeterminiziran ili oboje, predmeti opažanja postaju korisni.
  U oba slučaja, budući predmeti opažanja ne mogu biti predodređeni unaprijed i budući postupci agenta ovisit će o tim budućim predmetima opažanja.

Zaključak

- U nedeterminističkim okolinama agenti mogu primjeniti AND-OR stablo pretraživanja da bi stvorili potencijalne planove kojima dolaze do cilja neovisno o tome koji se ishodi pojavljuju tijekom provedbe.

Zaključak

- U nedeterminističkim okolinama agenti mogu primjeniti AND-OR stablo pretraživanja da bi stvorili potencijalne planove kojima dolaze do cilja neovisno o tome koji se ishodi pojavljuju tijekom provedbe.
  Takvo stablo sadrži dvije vrste čvorova: OR-čvorove, koji su naslijeđeni iz deterministčkog pretraživanja te AND-čvorove koji su karakteristični za nedeterminističko pretraživanje.

Zaključak

- U nedeterminističkim okolinama agenti mogu primjeniti AND-OR stablo pretraživanja da bi stvorili potencijalne planove kojima dolaze do cilja neovisno o tome koji se ishodi pojavljuju tijekom provedbe.
  Takvo stablo sadrži dvije vrste čvorova: OR-čvorove, koji su naslijeđeni iz deterministčkog pretraživanja te AND-čvorove koji su karakteristični za nedeterminističko pretraživanje.
- U nesigurnom svijetu vakuuma, gdje akcije pokreta ponekad dožive neuspjeh, idealno rješenje je ciklički plan.

LITERATURA


S. Russell,P. Norvig *Artificial Intelligence A Modern Approach Third Edition*


http://www.cs.nott.ac.uk/ nza/G52PAS/lecture6.pdf