PRIRODOSLOVNO-MATEMATIČKI FAKULTET
SVEUČILIŠTE U SPLITU

Ana Šuljug

Inteligentni agenti za pretraživanje Web-a

DIPLOMSKI RAD

SPLIT, studeni, 2008.

Studijska grupa: Matematika i informatika

Predmet: Primjena računala u nastavi

Inteligentni agenti za pretraživanje Web-a

DIPLOMSKI RAD

Student: Ana Šuljug

Mentor: prof.dr.sc. Slavomir Stankov

Neposredni voditelj: mr.sc. Ani Grubišić

SPLIT, studeni, 2008.

ZAHVALA

Veliko hvala prof. mr.sc. Ani Grubišić na velikom strpljenju, uloženom trudu i pomoći koju mi je pružila tijekom izrade ovog rada.

Također hvala i ostalima na podršci, pomoći i razumjevanju tijekom studiranja.

Sadržaj:

1. UVOD

Dobra vijest u vezi sa World Wide Web-om je što na njemu možemo pronaći skoro sve što nas zanima. A loša vijest je što to nije nimalo lak posao. Živimo u vremenu u kojem smo svakodnevno bombardirani velikom količinom informacija koje je nemoguće kvalitetno prihvatiti, pregledati, obraditi i iskoristiti na pravi način.

Ubrzanim rastom Web-a korisnici su često suočeni sa problemom preopterećenja informacijama (eng. Information overload). Taj problem se može okarakterizirati na dva načina: filtriranje informacija (eng. Information filtering – svakodnevno smo suočeni sa velikim brojem informacija (e-mail, novosti na Web-u,...) od kojih su samo pojedine relevantne i važne, problem se javlja zbog lošeg upravljanja dobivenim informacijama) i sakupljanje informacija (eng. Information gathering – informacije se neprikladno pohranjuju što otežava njihovo kasnije pronalaženje zbog preobilja drugih informacija) [JENN1998].

Web sadržaji brzo se mijenjaju i razvijaju, ali i zastarijevaju. Kako količina i dostupnost raznolikih informacija svakodnevno raste tako raste i potreba za što boljim snalaženjem s njima. Jedno rješenje ovog problema je primjena inteligentnih agenata za pretraživanje Web-a. Devedesetih godina prošlog stoljeća intezivno se radilo na području umjetne inteligencije, a u vezi tzv. inteligentnih agenata za pretraživanje Web-a.

Sve gore navedene činjenice bile su mi glavna motivacija da napišem ovaj rad. Drugo poglavlje započinje iznošenjem osnovnih osobina agenata. Tako je dan kratak pregled definicija agenata iz kojih proizlaze i različiti pristupi ovom području, osnovna svojstva inteligentnih agenata te elementarna arhitektura agenta. Potom se iznosi osnovni koncept rada pretraživačkih sustava kod kojih prikupljanje podataka obavljaju agenti za pretraživanje Web-a. Nakon toga upoznat ćemo se sa pojmom inteligentnog agenta za pretraživanje Web-a. U literaturi se ujedno koriste i nazivi Web pauci (eng. Web spider), Web puzači (eng. Web crawler) i Web roboti. Na kraju poglavlja dan je opis izgradnje jednostavnog inteligentnog agenta za pretraživanje Web-a.

Treće poglavlje opisuje metode koje agenti koriste za pretraživanje Web-a. Dana je osnovna podjela metoda na slijepe i usmjerene te njihove karakteristike. Navode se pretraživanja koja spadaju u slijepe metode te daje njihov opis i algoritmi pretraživanja. Za svaku pojedinu metodu pretraživanja dan je primjer kako bi se opisala sama tehnika pretraživanja. Potom se navode pretraživanja koja spadaju u usmjerene metode te se za svaku pojedinu metodu pretraživanja daje primjer i opis same tehnike pretraživanja.

U četvrtom poglavlju naveli smo nekoliko primjera inteligentnih agenata za pretraživanje Web-a. Dan je kratki opis pojedinih agenata, osnovne karakteristike i funkcije. Za pojedine primjere agenata navedene su i metode pretraživanja koje koriste, te linkovi za više informacija o njima.

2.INTELIGENTNI AGENTI ZA PRETRAŽIVANJE WEB-a

World Wide Web je postao nepregledan izvor informacija. Pronalaženje točno određene informacije je često vrlo teško zbog kompleksnosti organizacije i količine pohranjenih informacija. Informacije su pohranjene u dokumentima koji imaju svoja imena (web stranice) i nalaze se u određenim direktorijima. Pronalaženje određene informacije zahtijeva poznavanje adrese na kojoj je smještena, a pogoditi gdje se nalaze informacije koje nam trebaju je gotovo nemoguće. Srećom, pretraživački sustavi poput Google-a, Alta Viste i Yahoo-a, te mnogih drugih, nam olakšavaju pretraživanje Web-a. Mnogi pretraživački sustavi pretražuju web stranice te iz njih crpe podatke i dokumente kako bi zadovoljili korisnikove potrebe. Međutim, često rezultati tih pretraživanja nisu relevantni korisnikovom zahtjevu. Upravo da bi se izbjegao ovakav problem razvijeni su *inteligenti agenti za pretraživanje Web-a* (eng. Intelligent Web searching agent).

Inteligentni agenti imaju takve osobine i mogućnosti da se smatraju novim pomakom u informacijskoj znanosti. Koriste se širom svijeta, kako u akademskim zajednicama tako i u poslovnim krugovima kod donošenja bitnih poslovnih odluka.

Područje inteligentnih agenata je vrlo mlado. Istraživanja ovog područja mogu se podijeliti na razvijanje teorije o agentima, na razvijanje arhitekture agenata i na razvijanje jezika orijentiranih na agente (eng. agent oriented languages), kako onih u kojima su agenti napisani tako i onih koji služe za komunikaciju među agentima [BOŽI2001]. Unatoč preprekama koje postoje, inteligentni agenti predstavljaju izazov jer prednosti koje nude premašuju poteškoće koje stoje na putu njihove praktične primjene.

2.1. Općenito o agentima

Što je to agent? Općenito govoreći, *agent je entitet koji nešto radi i to radi u ime nekoga.* Definirati pojam agenta je podjednako teško kao i definirati pojam npr. objekt. Pojam je jednostavno preopćenit. Postoje i mnogo određenije definicije što su agenti. Međutim, sve definicije odražavaju raznolikost shvaćanja pojma, što je posljedica mladosti čitavog područja.

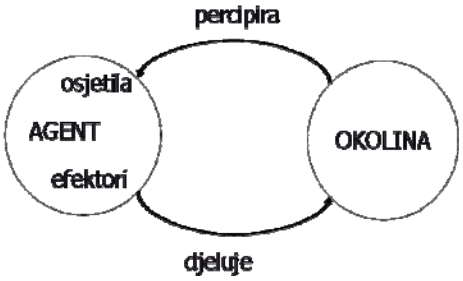
Intuitivno se pri spomenu agenta podrazumijeva objekt koji samostalno djeluje u nekoj okolini, prilagođava se okolini u kojoj djeluje, ima sposobnost percipiranja stanja u kojoj se njegova okolina nalazi, djelovanjem mijenja stanja okoline i ima sposobnost učenja [ROSI2000].

2.1.1. Definicije agenata

Promotrimo nekoliko definicija agenata:

- Pojam agent prikazuje dva ortogonalna koncepta:
  (i) sposobnost agenta da djeluje autonomno tj. neovisno od korisnika
  (ii) sposobnost agenta da zaključuje o nekom području interesa (MuBot agent prema [FRAN1996].
- Agent je nešto što može prihvaćati vlastitu okolinu kroz senzore i djelovati na okolinu kroz efektore (AIMA agent prema [FRAN1996].
- Autonomni agenti su sustavi koji prihvaćaju kompleksnu dinamičnu okolinu, osjećaju je i djeluju autonomno u toj okolini te na taj način realiziraju skup ciljeva zbog kojih su oblikovani (Maes agent prema [FRAN1996].
- Agenti su stalni programski entiteti namjenjeni za posebnu svrhu. To znači da se agenti razlikuju od potprograma; agenti imaju svoje vlastite ideje o izvršenju zadatka, svoj vlastiti raspored. Posebna svrha razlikuje agente od potpuno višefunkcionalnih aplikacija; agenti su manji (KidSim agent prema [FRAN1996].

Rad agenta razmatra se kroz njegovo djelovanje unutar dodijeljene mu okoline. Interakcijom agenta s okolinom mijenjat će se stanja okoline. Slika1 opisuje interakcija agenta i okoline: agenti pomoću senzora (osjetila) percipiraju stanja okoline te ih mijenjaju preko aktuatora (efektora), tj. djeluju na okolinu.



Slika 1. Interakcija agenta s okolinom (modificirano prema [RUSS1995])

Opće je prihvaćena definicija agenata prema kojoj se agenti mogu podijeliti na slabu i jaku predodžbu [WOOL1995].
Slaba predodžba agenata ima slijedeća obilježja:

- *Samostalnost* (eng. autonomous): Agent ima sposobnost izvršavati operacije samostalno bez potrebe za intervencijom korisnika. Ovo svojstvo zahtijeva od agenta posjedovanje mehanizama koji će mu omogućiti samostalno dohvaćanje svih potrebnih informacija o okruženju u kojem se nalazi. Ovisno o dobivenim informacijama agent izvršava pripadajuće operacije. U slučaju greške mehanizmi moraju omogućiti nastavak rada agenta
- *Društvenost* (eng. social ability): Agent mora posjedovati mehanizme koji mu omogućuju koordiniranje svojih operacija s drugim agentima u svojem okruženju. Obilježje društvenosti agenta ostvareno je korištenjem komunikacijskih protokola, interakcijskih protokola i ontologija.

- *Reaktivnost* (eng. reactivity): Reaktivni agenti ne posjeduju interni model koji im omogućava predviđanje budućeg stanje okoline, već reagiraju isključivo na temelju sadašnjeg znanja i skupa akcija koje poznaju. Oni reagiraju na utjecaje iz okoline s akcijom predviđenom za trenutno stanje okoline i vrstu utjecaja.
- *Proaktivnost* (eng. proactiveness): Za razliku od reaktivnih agenata, koji samo reagiraju na utjecaje iz okoline, proaktivni agenti posjeduju mehanizme koji im omogućavaju utjecanje na okolinu mijenjajući pritom njeno stanje. Pri tome iskazuju ciljno orijentirano ponašanje, odnosno, svaki agent ima jedan ili nekoliko ciljeva. U situacijama kada stanje okoline nije u skladu s njegovim ciljevima, agent utječe na okolinu dok ne ostvari zadani cilj

U području umjetne inteligencije vrlo je često zastupljena jaka predodžba agenta koja ima slijedeća obilježja:
- *Pokretljivost* (eng. mobile): Agent se može kretati s jednog mrežnog čvora na drugi. Svaki pokretni agent sastoji se od tri komponente: (i) programskog koda koji sadrži logiku agenta, (ii) podataka, odnosno, internih atributa koji predstavljaju znanje koje agent posjeduje i (iii) stanje izvođenja
- *Racionalnost* (eng. rationality): definira da ako agent ima skup ciljeva, od kojih je samo jedan aktivan, on neće izvoditi akcije koje bi mogle biti u suprotnosti s njegovim trenutnim ciljevima. Racionalan agent uvijek mora izvoditi akcije koje bi u konačnici maksimizirale očekivani rezultat pri tome koristeći svoje znanje o trenutnom i budućem stanju okoline
- *Dobronamjernost* (eng. benevolence): Agentovi ciljevi ne smiju biti međusobno konfliktni, ako se od agenta želi da maksimizira očekivani rezultat.

Takvi agenti poprimaju i ljudske karakteristike kao što su emocije ili koriste psihološke osobine ljudi.

2.1.2. Inteligentni agenti

Često se umjesto pojma agent koristi pojam inteligentni agent čime se naglašava da agent mora posjedovati i određenu razinu inteligentnog ponašanja pri obavljanju svojih zadataka. Definicija inteligencije je sigurno složenija od definicije agenta. Inteligencija se opisuje različitim atributima poput sposobnosti učenja, razumijevanja, planiranja i predviđanja, rješavanja problema, sintetiziranja novih ideja i modeliranja vanjskog svijeta. Inteligenciju možemo opisati kao posjedovanje mehanizama ponašanja usmjerenih rješavanju zadanog cilja [ROSI2000].

Inteligentni agenti sadrže određeni stupanj inteligencije (nivo zaključivanja i učenja odnosno sposobnosti da prihvate korisnikove izjave o ciljevima i da obave zadatke koji se postave pred njih). Inteligentni agenti su agenti koji prate naše ponašanje i na temelju njega su u stanju naučiti kako mi rješavamo probleme i sl. Kad su to naučili oni mogu te zadatke obavljati samostalno, uz povremene provjere vlasnika kod nejasnih ili novih situacija [BOŽI2001].

Svojstva inteligentnih agenata [RUDO2004]:
- učenje – od korisnika, od drugih agenata, iz ostalih izvora
- suradnja – radi s drugim agentima kako bi postigao svoj cilj
- mobilnost – pokretljivost agenata po mreži, izvođenje na različitim računalima
- personalizacija – poznavanje svog korisnika, njegovih interese i preferencija

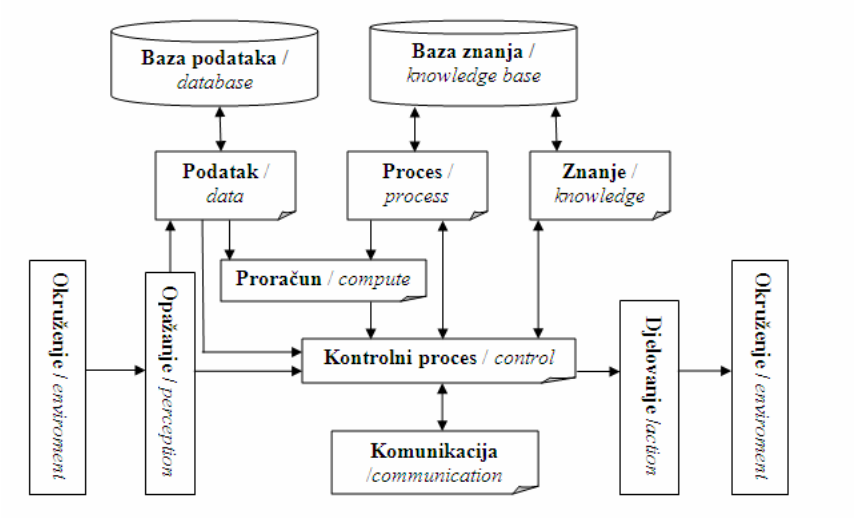
- adaptibilnost – uče iz različitih izvora, te iz korisničkih akcija.

2.1.3. Arhitektura agenata

Kad se govori o *arhitekturi agenata* misli se na takvu konstrukciju računalnog sustava koji omogućuju agentima da to zaista i budu (da imaju osobine koje ih čine agentima). Arhitektura agenata je određena metodologija za izgradnju agenata, ona određuje kako se agent dekomponira u niz modula i kako su ti moduli povezani. Ona pokazuje i način na koji moduli obavljaju svoje aktivnosti, a u svrhu obavljanja zadatka. U arhitekturi agenata mogu se i pokazati veze između različitih agenata, a ne samo struktura jednog agenta. Kod arhitekture ne može se govoriti o jednoj, jedinstvenoj arhitekturi agenta. Kakva će ona biti ovisi o svrsi agenta. Postoje određeni zahtjevi koji se postavljaju pred arhitekturu agenata. Prije svega ona bi trebala omogućiti autonomiju, komunikacije, učenje, orijentiranost ka cilju, mobilnost i stabilnost. Postoje različite razine arhitekture u smislu da određeni agent ne mora imati sve osobine, ali one moraju biti iz gore navedenog skupa kako bi se entitet smatrao agentom. Jedan od najčešćih zahtjeva za arhitekturu jest da omogući samostalnost. Učenje i zaključivanje drugi su važan aspekt koji mora osigurati arhitektura agenta. Budući da agent djeluje u okolini koja sadrži i druge agente i različite usluge, potrebna je takva arhitektura koja standardizira komunikaciju među agentima i omogućuje pristup određenim uslugama[BOŽI2001].

Arhitektura agenta je zapravo struktura njegova programa (Slika2). Model samostalnog agenta (u smislu ostvarenja cilja) sastoji se od 8 jedinica i uključuje (Shen prema [RAJŠ2007]):

- jedinicu za opažanje (eng. perception unit),
- jedinicu za obradu (eng. process unit),
- kontrolnu jedinicu (eng. control unit),
- djelatnu jedinicu (eng. action unit),



Slika 2. Arhitektura agenta (Shen prema [RAJŠ2007])

- jedinicu za komunikaciju (eng. communication unit),
- jedinicu znanja (eng. knowledge unit),
- računsku jedinicu (eng. compute unit),
- jedinicu sa podacima (eng. data unit).

Jedinica za opažanje očitava podatke iz agentovog okruženja. Ona sadrži listu pojedinačnih stanja koja definiraju ukupno stanje okruženja. Ukoliko se neko od pojedinih stanja promijeni, lista podataka se ažurira te se obavještava kontrolnu jedinicu o promjenama kako bi poduzela odgovarajuće mjere. *Jedinica za obradu* sadrži cilj ili više ciljeva te njihove međusobne odnose. *Djelatna jedinica* sadrži sve djelatnosti koje je agent sposoban obaviti. *Kontrolna jedinica* odlučuje koje će se djelatnosti obaviti radi ostvarenja cilja kojeg također ona odabire. Ukoliko jedinica za opažanje uoći promjene u okruženju, tada kontrolna jedinica aktivira reakcijska djelovanja na te promjene. U *računskoj jedinici* su definirane funkcije kojima se odabiru ciljevi i optimalna djelovanja. Također sadrži algoritme i mehanizme odabiranja optimalnih djelovanja. Kontrolna jedinica poziva funkcije iz računske jedinice kako bi se odredio slijedeći cilj i prikladna djelovanja. Jedinica za razumijevanje nadgleda dio odgovoran za znanje agenta koje se primjenjuje pri rješavanju stvarnih problema. *Jedinica s podacima* omogućava pristup svim mehanizmima u bazu podataka. *Komunikacijska jedinica* definira načine komunikacije između agenata.

Danas se inteligentni agenti koriste pri obavljanju mnogih zadataka kao što su pretraživanja sadržaja na Web-u, razvrstavanja elektroničke pošte, odabir korisniku zanimljivih priloga u novinskim grupama, podrška računalnim mrežama, pretraživanje baza podataka i prilagodbe korisničkog sučelja raznim individualnim potebama korisnika.

2.2. Agenti za pretraživanje Web-a

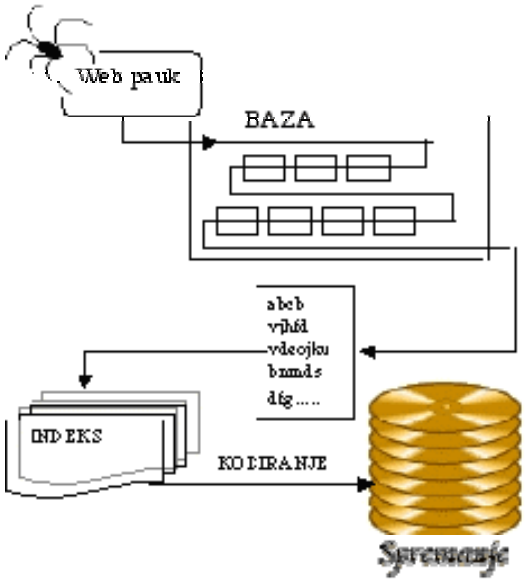
Porast popularnosti i broja dostupnih sadržaja, koji su u potpunosti neorganizirani, ruši osnovni koncept Weba - jednostavno pronalaženje informacija. Zbog toga se počevši od 1994. g. razvija niz alata, čija je jedina zadaća pronalaženje željenih sadržaja na Web-u. Ti alati se razvijaju u dva osnovna smjera, pa razlikujemo *kataloge* i *pretraživače* (eng. search engines). Jedni i drugi sadrže bazu web stranica, indexa, URL-ova (eng. Uniform Resource Locator - adresa koja jedinstveno ukazuje na neki sadržaj na Internetu), a osnovna razlika je u načinu prikupljanja tih podataka. Kod kataloga, sadržaj baze isključivo ovisi o ljudima, tj. ljudi pretražuju Web, za interesantne stranice napišu sažetak i sažetak se uz URL adresu stranice sprema u bazu. Katalozi se koriste još i danas, a jedan od najpopularnijih je Yahoo. Za razliku od kataloga, prikupljanje podataka kod pretraživača je u potpunosti automatizirano i obavljaju ga *agenti za pretraživanje Web-a* (eng. Web Searching Agent) koji su još poznati pod nazivom *pauci, puzači, crvi* (eng. spider, crawler, worm) [LARI2005].

Agenti za pretraživanje Web-a „lutaju“ Web-om u potrazi za novim stranicama, i kada ih pronađu „dovlače“ ih na računalo i pohranjuju u bazu. Mada razni nazivi poput pauka, lutalica, puzača, crva stvaraju privid da se agenti gibaju, to nije istina, oni su stacionirani na računalu i na to računalo „dovlače“ stranice. Oni „skaču“ s Web stranice na Web mjesto preko poveznica,

prikupljajući naslove svih mjesta, URL, i najmanje od njihovih tekstovnih sadržaja. Kada nađu mjesto, oni pregledavaju (eng. scan) Web stranice toga mjesta i zapisuju (eng. record) sve informacije u indeks [TEŽA2001].

U slučaju agenata koji pretražuju Web, agentova okolina je Web, a komunikaciju sa korisnikom omogućuje računalni terminal. Ono što agent percipira su riječi HTML dokumenta naučene korištenjem programskih detektora (senzora) povezanih kroz cijelu mrežu (Internet) uz pomoć HTTP-a. Zadatak agenta je ustanoviti je li postignut cilj, tj. je li pronađena Web stranica koja sadrži ključnu riječ ili frazu, i ako nije, pronaći druga mjesta koja će posjetiti i potražiti u svrhu postignuća cilja, tj. obavljanja zadatka. Agent djeluje na okolinu koristeći izlazne metode kako bi obavijestio korisnika o statusu pretraživanja ili krajnjim rezultatima, koji bi trebali predstavljati postignut cilj [YOUN1999].

Rad pretraživača mogao bi se shematizirati na sljedeći način. Web agent pretražuje bazu



Slika 3. Rad pretraživača [TEŽA2001]

podataka, izlistava riječi koje zadovoljavaju upit (ključnu riječ) i zapisuje mjesto gdje ih je našao, indeksira pronađene riječi u indeks prema vlastitom sustavu prosudbe, kodira podatke na sigurno mjesto i sprema podatke za uporabu (Slika3).

Indeksiranjem (sastavljanje liste dokumenata na način da sa svake stranice koju pronađe sakuplja ključne riječi i ubacuje ih u indeks riječi) se izvlače informacije iz dokumenata i sprema ih u svoju bazu podataka. Cilj je posjetiti milijune Web mjesta i ostati s njima u vezi koliko god je moguće. U svojemu hodu po mreži razni pretraživači međusobno dijele više baza podataka. Što se sprema u bazu podataka, ovisi o određenom pretraživaču. Neki indeksiraju svaku riječ iz Web dokumenata, a drugi indeksiraju samo naslov. Kada se provodi pretraživanje preko ključne riječi ili fraze, pretražuje se cijela baza podataka, ali u rezultatu se pokazuju samo Web stranice u kojima je nađena ta ključna riječ ili fraza. Baza podataka sadrži prikupljene stranice, a pretraživački program na osnovu upita pretražuje bazu podataka, rangira dobivene rezultate i konačno ih šalje korisniku. Uz prikupljane novih Web stranica agenti se još mogu koristiti i za

osvježavanje sadržaja u bazi. Osim potrage za novim stranicama, agenti se u pravilnim vremenskim razmacima, najčešće mjesečno, vraćaju na lokacije koje su ranije posjetili i provjeravaju da li je u međuvremenu sadržaj Web stranica promijenjen i ako je onda izmjene dodaju u svoje indekse [TEŽA2001].

Iako su pretraživački mehanizmi dosta uspješni, ipak se milijunima korisnika događa da klikaju po linkovima koje dobiju kao rezultat pretraživanja i ne dođu do Web stranice s traženom informacijom. Pretraživački mehanizam često pronalazi ono što korisnik želi, ali ne uvijek. Da bi se izbjegli ovakvi probleme razvijeni su posebni računalni programi koji djeluju u ime korisnika, te sadrže određeni stupanj inteligencije. Ti računalni programi nazivaju se *inteligentni agenti za pretraživanje Web-a* (eng. Intelligent Web searching agent).

2.3. Inteligentni agenti za pretraživanje Web-a

Kao što smo već rekli u uvodnom dijelu, izražena dinamičnost i velika distribuiranost podataka zahtjeva od računalnih programa ne samo da odgovaraju na zahtjeve, nego da i inteligentno sudjeluju u aktivnostima u svojem okruženju, te se kontinuirano prilagođavaju trenutnom stanju svoje okoline i na taj način aktivno traže način kako da što bolje ostvare interese korisnika. Važnost informacija je danas neupitna, no još važnije je prave informacije imati u pravo vrijeme. Koliko smo samo puta kao rezultat pretraživanja dobili niz podataka koji nisu ispunili naša očekivanja.

Inteligentnim agentima za pretraživanje Web-a nazivaju se računalni programi što samostalno izvode neki pretraživački posao "u ime i za račun" korisnika. Smješteni su u računalu vlasnika, što ne mora nužno biti (a najčešće i nije) računalo krajnjeg korisnika, već neko internetsko web mjesto. Korisnik ih mora "napuniti" informacijama o područjima svojega interesa, pravilima pretraživanja, prioritetima, te eventualnim vremenskim ograničenjima. Nakon što agent obavi postavljeni mu zadatak, analiziraju se rezultati, a agent ispravlja sam sebe ako ti rezultati nisu zadovoljavajući prema nekom unaprijed postavljenom kriteriju [PANI2000].

Inteligentni agenti za pretraživanje Web-a mogu imati različite stupnjeve samostalnosti u izvršavanju zadataka u odnosu prema korisniku i njegovim potrebama. Tako, primjerice, neki od njih mogu "samo" prikupljati informacije, neki drugi filtrirati (selektirati) poruke primljene elektroničkom poštom ili putem dostavnih lista, a neki surađivati s drugim agentima, te obavljati vrlo složene i specifične zadatke. Razvijene su tri vrste takvih agenata [PANI2000]:

- Web puzači (eng. Web crawler),
  Pokušavaju korisniku dati cjelovit pregled informacijske ponude, "šetajući" Web-om i izvještavajući korisnika o onome što su pronašli.
- Web pauci (eng. Web spider),
  To su programi koji se pomoću "crvića" (eng. worm) "uvlače" u dubinu hipermedijskih dokumenata, odnosno web stranica, te pronađene informacijske sadržaje indeksiraju i pohranjuju u vlastite baze podataka, koju korisnik kasnije može jednostavno lokalno (na svome računalu) pretraživati i analizirati.

- Web roboti (eng. Web robot).
  To su programi koji mogu gotovo potpuno nezavisno od korisnika izvršavati kompletne transakcije, poput kupovine na daljinu, rezervacije zrakoplovnih karata, novčanih transakcija itd., u skladu s naputcima koje im ih je korisnik jednom ranije dao.

Područje primjene inteligentih agenata za pretraživanje je vrlo široko. Tako se inteligentni agenti koriste i za :

- *statističke analize* - agenti mogu računati prosjećan broj dokumenata po serveru, odnos različitih tipova datoteka, prosjećan broj Web stranica, broj Web servera na Internetu…
- *osvježavanje URL adresa* - glavni problema kod održavanja Web stranica je u tome što adrese na druge stranice mogu postati nevažeće, tzv. 'mrtve veze', ako se ta stranica premjesti ili čak ukloni. Stoga su razvijeni agenti, čiji je zadatak provjera ispravnosti adresa.
- *zrcaljenje* (eng. mirroring) - je postupak preslikavanja sadržaja Web servera na drugo računalo s ciljem smanjenjem prometa na nekom dijelu mreže. Agenti se mogu koristiti za uspostavljanje zrcalne slike Web stranica, ali ova metoda još nije u potpunosti razvijena tako da kad dođe do promjene neke Web stranice moraju se osvježavati sve zrcalne stranice, a ne samo one koje su promijenjene.
- *indeksiranje* - prikupljanje ključnih riječi iz Web stranica kako bi pretraživanje bilo što lakše i brže. Najčešće se agenti koriste za prikupljanje naslova i nekoliko prvih paragrafa teksta sa stranice.
- *pronalaženje podataka* - inteligentni Web agenti imaju veliki potencijal u pronalaženju podataka (eng. data mining - proces pronalaženja uzorka u velikoj količini podataka kroz niz pretraživanja), uz to mogu donositi odluke zasnovane na prethodnim pretraživanjima, da bi obavili što kompleksnije pretraživanje.
- *kombinirana upotreba* - jedan agent može izvršavati i više spomenutih zadataka [LARI2005].

2.3.1. Izgradnja jednostavnog inteligentnog agenta za pretraživanje Web-a

U ovom dijelu pokazat ćemo kako izgraditi jednostavnog inteligentnog agenta za pretraživanje Web-a [YOUN1999]. Parametri koje trebamo definirati za agenta su:

- *cilj* – ključna riječ ili fraza,
- *početna lokacija* – URL adresa od koje počinje pretraživati,
- *broj iteracija* – koliko ponavljanja (što više URL-ova to bolje),
- *vremensko ograničenje* – koliko dugo će pretraživati,
- *metoda pretraživanja* – na koji način će birati put.

Razvoj inteligentnog agenta podrazumijeva 4 glavne faze:

- *Faza inicijalizacije (eng. initialization)* uključuje postavljanje svih varijabli, struktura i polja, te unošenje ulaznih informacija o pretraživanju: ključna riječ, cilj pretraživanja, broj Web stranica koje sadrže ključnu riječ (pronađen u ograničenom vremenu), početna lokacija, metoda pretraživanja (ukoliko ih agent podržava više).

- *Faza percepcije (eng. perception)* je usmjerena na korištenje znanja za kontaktiranje Web stranice i pronalaženje informacija s te lokacije. Treba identificirati prisutnost ključne riječi i linkove s te stranice na druge.
- *Faza akcije (eng. action)* – tijekom ove faze agent uzima sve prikupljene informacije i odlučuje je li cilj postignut tj. je li meta pronađena i pretraživanje gotovo. Ukoliko cilj nije postignut mora odlučiti koju adresu će sljedeću posjetiti. Ovaj dio odnosi se na inteligenciju agenta, a koliko će pametan biti ovisi o metodi pretraživanja koju će izabrati.
- *Faza efekta (eng. effect)* – u ovoj fazi agent izlistava Web adrese na kojima je pronašao ključnu riječ, daje status i povratnu informaciju korisniku.

Tijekom *faze inicijalizacije* stvara se struktura podataka koja sadrži informacije o posjećenim lokacijama. Za svako pretraživanje moramo jasno definirati cilj i ustanoviti metu. Za početak, radi jednostavnosti, počnemo s pretraživanjem po ključnoj riječi. Korisnik će sigurno željeti pogledati više od samo jedne pronađene Web stranice s ključnom riječi, pa bi broj stranica također trebao biti specificiran kao cilj. Kako korisnik želi povratnu informaciju što je brže moguće, treba postići ciljani broj stranica u najkraćem mogućem vremenu. Ukoliko je agent fleksibilan pa dozvoljava odabir metode pretraživanja, treba ustanoviti koju od ponuđenih metoda upotrijebiti. Karakteristike pojedinih metoda predstavljaju ponašanje agenta. Program može biti postavljen tako da pita korisnika kakvog agenta želi: agresivnog (eng. aggressive), umjerenog (eng. moderate) ili prigodnog (eng. casual). Mogu se primijeniti i razne kombinacije i permutacije karakteristika pretraživanja. Agent može započeti s pretraživanjem umjerenim tempom i, ovisno o rezultatima pretraživanja, prijeći u agresivan mod. Nakon inicijalizacije i definiranja metode pretraživanja, počinje samo pretraživanje.

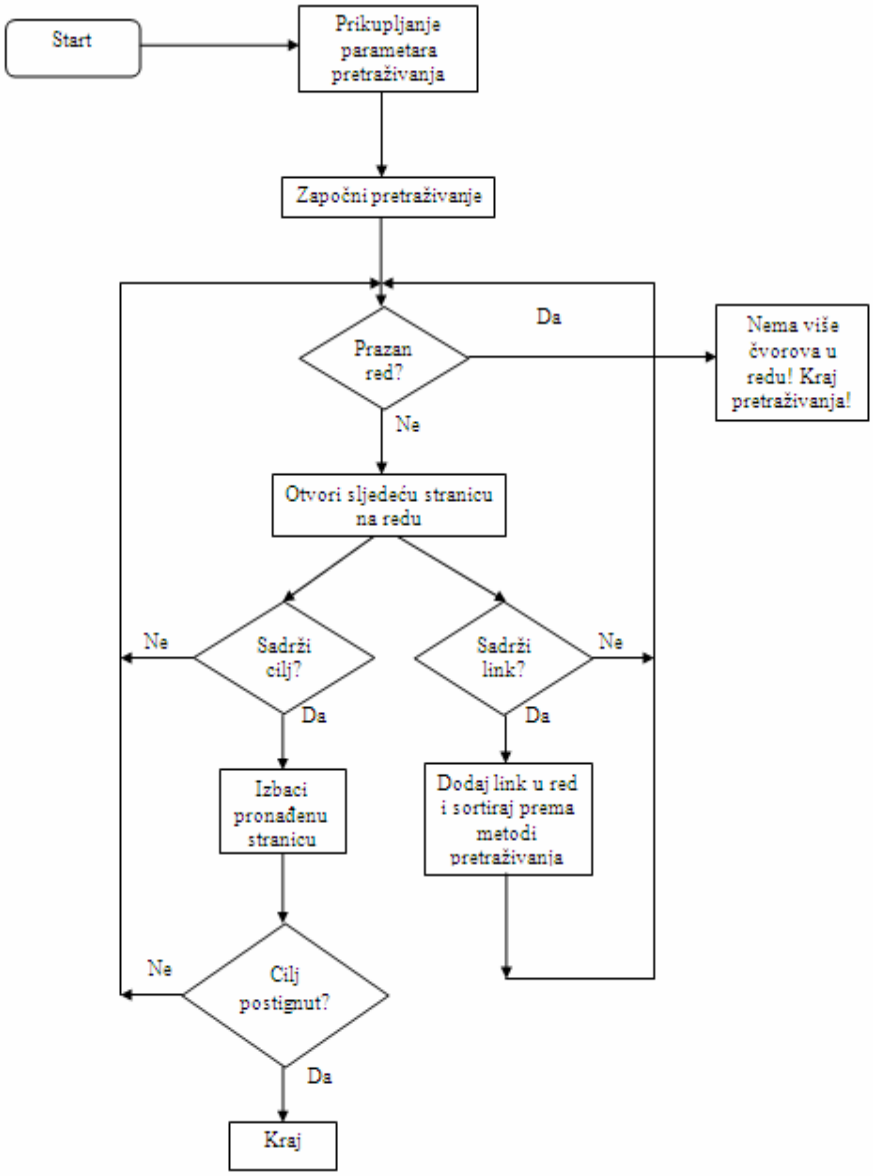
Web agent ulazi u *fazu percepcije* otvaranjem komunikacijskog mehanizma (eng. socket-a) na HTTP portu za specificiranu Web stranicu. Nakon uspostavljanja veze, HTTP naredba GET zajedno s informacijom o direktoriju poziva server da ispostavi zatraženu stranicu. Tada se dobivena stranica gramatički analizira radi pronalaženja linkova, koji se onda poredaju u čvorove kako bi se moglo ustanoviti je li prisutna tražena meta. Ključni HTML tagovi koji ukazuju na linkove su "HREF=" i "SRC=", a link u sebi mora sadržavati "http" negdje u tekstu. Ukoliko tražimo neki drugi tip podataka možemo napraviti takvog agenta da traži npr. slike ("JPG" ili "GIF" reference) ili bilo koji tip dokumenta koji se lako prepoznaje. Mora se paziti da se od agenta ne zahtjevaju tipovi dokumenata kojima on ne zna rukovati. Web stranice mogu sadržavati različit broj hyperlinkova ili referenciranih HTML stranica - počevši od nijednog pa do stotina ili čak tisuća. Pametno bi bilo postaviti gornju granicu broja linija koje će agent primiti. Gramatička analiza teksta može biti programska zadaća pogotovo ako se radi u C-u, ali korištenjem napredne knjižnice ili nekih oblika jezika za gramatičku analizu teksta, zadatak se može znatno pojednostavniti.

Nakon što je stranica prihvaćena i klasificirana, slijedi *faza akcije*. Kako je agent uzeo Web stranicu i izvadio iz nje važne informacije, sada je vrijeme za malo "mišljenja" i akcije. Ako je cilj postignut, tj. pronađen je ciljani broj Web stranica onda je pretraživanje završeno i bilo je uspješno. Ukoliko ciljani broj Web stranica nije postignut, slijedi donošenje odluke koji je sljedeći korak. Središte odluke je izbor algoritma pretraživanja. U red se stavljaju čvorovi (linkovi) iz tekućeg dokumenta. Strategija pretraživanja određuje kojim redoslijedom će se oni posjećivati i na koji način. Što je strategija pretraživanja informiranija, to je veća vjerojatnost da će cilj biti postignut. Slijepa pretraživanja obično ne mare za dodatne informacije, jedino mogu zatražiti informaciju o cijeni koja je za Web stranice određena kao njihova udaljenost od Web

agenta. Udaljenost, ili brzina pristupa, može se koristiti kao heuristička informacija budući da se svaki čvor može promatrati kao dio cilja, a dio cilja je minimizirati ukupno vrijeme pretraživanja. Osim brzine pristupa heuristička informacija može biti i zemljopisna udaljenost tj. zemlja kojoj Web stranica pripada. Tako je npr. većina Web stranica u SAD-u u domenama "COM", "NET", "EDU", "ORG", "GOV", "MIL" ili "US" pa agent iz SAD-a može stranicama koje nisu u ovim domenama dodijeliti veću cijenu jer nisu iz njegove zemlje i vjerojatno su na drugom jeziku. Ova heuristika je prilično slaba jer mnoge institucije izvan SAD-a imaju "COM" ili "EDU" adrese (kompanije ili obrazovne institucije). Prema dogovoru kako se dodjeljuju IP adrese i imena Web stranicama, ne postoji sasvim točan način kako odrediti lokaciju Web stranice. Ne znatno olakšanje snalaženja u ovom kaosu daje CIDR (Classless Inter-Domain Routing) zapis IP adrese, odnosno besklasno adresiranje koje uz običnu adresu nosi informaciju o IP prefiksu koji predstavlja broj bitova koji pripadaju mrežnom dijelu adrese (mrežna maska), ali s obzirom na broj IP adresa, pomoć je zaista minimalna.
Nakon što metoda pretraživanja doda ili izbriše čvorove može ih sortirati po cijeni i/ili heuristici. Izbor algoritma sortiranja nije toliko važan, ali naravno da će onaj koji radi brzo i točno poboljšati performanse agenta i skratiti vrijeme pretraživanja. Ukoliko je prostor pretraživanja iscrpljen i nema više čvorova u redu agent može dobiti novu početnu vrijednost i početi odatle ili poslati status i završiti s radom. O strategijama pretraživanja raspravljat ćemo u sljedećem poglavlju.

U *fazi efekta* Web agent izvještava korisnika o statusu pretraživanja i pokazuje mu rezultate. Efektor iz akcije zasnovan na percepciji okoline mora dati povratnu informaciju korisniku.

Web agent se premješta iz inicijalne faze u petlju koja se sastoji od faza percepcije, akcije i efekta, sve dok se cilj pretraživanja ne postigne ili dok se ustanovi da se ne može pronaći traženo. Slika 4. ilustrira redoslijed osnovnih akcija prijelaza iz faze u fazu.



Slika 4. Niz osnovnih akcija inteligentnog Web agenta (modificirano prema [YOUN1999])

3. METODE PRETRAŽIVANJA WEB-a

U području umjetne inteligencije, rješavanje problema se može okarakterizirati kao sistematično pretraživanje mogućih ishoda u svrhu pronalaska rješenja. Rješavanje problema putem *algoritama za pretraživanje* (eng. search algorithm) je vrlo česta tehnika. U računalnoj znanosti algoritam za pretraživanje u širem kontekstu je algoritam koji daje rješenje nakon vrednovanja skupa mogućih rješenja. Skup svih mogućih rješenja problema naziva se *prostor pretraživanja* (eng. search space) [WIKI2008].

Zadatak agenta je doći iz *početnog stanja* (eng. initial state) do *ciljnog stanja* (eng. goal state), koje je rješenje problema, slijedom događaja koji uzrokuju promjene stanja. Čest je slučaj da skup stanja između početnog i ciljnog nije jednoznačno određen, nego postoji više različitih, često i različito učinkovitih puteva od početnog do krajnjeg stanja. Za rad takvih sustava potrebno je ugraditi algoritme za traženje spomenutih puteva. Otežavajuća okolnost kod izrade spomenutih algoritama je nepostojanje formalnih postupaka koji bi na temelju početnog i krajnjeg stanja sustava određivali skup stanja na putu između njih. U takvim je slučajevima potrebno pribjeći metodi *pretraživanja prostora stanja* [DALB2001].

Pretraživanje prostora stanja je postupak koji slijedno prolazi kroz stanja u kojima bi se mogao nalaziti sustav, te uspoređivanjem *trenutnog stanja* (eng. current state) s *ciljnim stanjem* utvrđuje da li je postupak došao do kraja. Skup stanja organiziran je u strukturu grafa ili stabla pa se pretraživanje svodi na pronalaženje puta kroz graf ili stablo. Čvorovi grafa pritom predstavljaju stanja sustava, a dva su čvora povezana granom u grafu ukoliko se iz jednog stanja može prijeći u drugo. Osim toga, svaki čvor ima vezu na čvor prethodnik tako da se može pronaći put u grafu kad se pronađe konačno stanje. Put se gradi tako se krene od ciljnog čvora, te se slijedeći veze na prethodnike, dođe do početnog čvora grafa [DALB2001].

Problem, općenito, može imati i složeniju strukturu. Postoji cijeli izbor mogućih događaja od kojih nitko ne garantira da će nas upravo taj dovesti do rješenja. Opći pristup rješavanja takvog procesa jest *generirati i ispitati* (eng. generate and test), tj. primjeniti neku akciju kako bi proizveli novo stanje, a zatim ispitati je li to stanje ciljno stanje i ako nije ponoviti proceduru. Procedura ponavljanja procesa generiranja i ispitavanja jest sam *proces pretraživanja*. Pretraživanje može biti kratko ili dugo, ovisno o složenosti problema i učinkovitosti agentove strategije rješavanja problema. U nekim slučajevima određeno rješenje je bolje od drugog rješenja iako oba predstavljaju konačno rješenje, tj. cilj. Tada se uvodi pojam *vrijednost (cijena) putanje* (eng. path cost) koja predstavlja zbroj vrijednosti pojedinih koraka putanja [HEYL1998].

Agent generira sve moguće ishode nekog događaja, bira one koji mu odgovaraju za pronalaženje navedenog cilja (rješenja) i izvršava niz koraka počevši od početnog (inicijalnog) stanja kako bi stigao do cilja. U slučaju inteligentnog agenta za pretraživanje radi se o korištenju algoritama pretraživanja za "navigaciju" kroz World Wide Web kako bi agent došao do cilja, tj. pronašao ono što mu je zadano.

Razlikujemo dvije osnovne klase algoritama za pretraživanje: *slijepo* ili *neinformirano pretraživanje* (eng. blind search or uninformed search) i *heurističko* ili *usmjereno (informirano) pretraživanje* (eng. heuristic or informed search) [YOUN1999].

Slijepo pretraživanje je pretraživanje koje nema nikakvih informacija o broju koraka ili vrijednosti putanje (vrijednost udaljenosti čvora od početnog čvora) od početnog do krajnjeg stanja, tj. cilja. U ova pretraživanja spadaju:

- pretraživanja po dubini (eng. Depth-first search)
- pretraživanja po širini (eng. Breadth-first search)
- pretraživanja s jednolikom cijenom (eng. Uniform-cost search)
- pretraživanje do određene dubine (eng. Depth-limiting search)
- iterativno pretraživanje po dubini (eng. Iterative deeping search)

Heurističko pretraživanje je pretraživanje koje ima dodatne informacije o cilju, cijenu putanje ili broj koraka. Te informacije čine ova pretraživanja boljim od slijepih te im omogućuju gotovo racionalno ponašanje. U ova pretraživanja spadaju:

- pretraživanje najboljim prvim (eng. Best first search)
- pretraživanje penjanjem (eng. Hill-climbing search)
- A* pretraživanje (eng. A* search)
- ograničeno pretraživanje po širini (eng. Beam search)
- IDA* pretraživanje (eng. Iterative deeping A* search)

Metode pretraživanja razmatraju se u okviru ovih kriterija [FRIS2007]:

- Potpunost (eng. completeness): da li strategija pretraživanja garantira pronalazak rješenja?
- Vremenska složenost (eng. time complexity): koliko je vremena potrebno za pronalazak rješenja?
- Složenost prostora (eng. space complexity): koliko je memorije potrebno za izvođenje pretraživanja?
- Optimalnost (eng. optimality): da li strategija pretraživanja pronalazi visoko kvalitetno rješenje među više rješenja?

Vremenska i prostorna složenost se mjere u terminima [FRIS2007]:

- b: najveći faktor grananja stabla pretraživanja
- d: dubina najkraćeg puta do rješenja (korijen je dubine 0)
- m: najveća dubina stabla pretraživanja (može biti beskonačno)

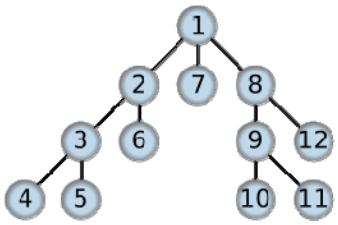
3.1. SLIJEPE METODE PRETRAŽIVANJA

Slijepo ili neinformirano pretraživanje je pretraživanje u kojem su jedine raspoložive informacije početno stanje, dozvoljene operacije nad stanjima i test na rješenje problema. Strategija slijepog pretraživanja ne uzima u obzir specifičnu prirodu problema. Na taj način slijepi algoritmi pretraživanja mogu se poopćiti, odnosno na isti način mogu se primjeniti na široko područje problema. Međutim većina prostora pretraživanja je vrlo velika, te ovakvi algoritmi nisu efikasni osim za male primjere. Slijepo pretraživanje napreduje sistematski kroz prostor pretraživanja

pretražujući čvorove u nekom prethodno definiranom redoslijedu ili birajući ih slučajno. Slijepe metode pretraživanja se razlikuju međusobno po načinu obilaženja čvorova [PAVK2005].

3.1.1. Pretraživanje po dubini

Pretraživanje po dubini (eng. depth first search, DFS) je način pretraživanja u kojem se koristi stog kao struktura podataka u koju ćemo pohranjivati one vrhove grafa koje smo posjetili. Stog radi na principu LIFO (eng. last in first out), tj. zadnji podatak koji je stavljen na stog će prvi biti obrisan. Samo pretraživanje započinje od proizvoljnog vrha u grafu kojeg označavamo kao početnog. Zatim, nastavljamo sa susjednim vrhom koji mora zadovoljavati neki od zadanih kriterija, npr. abecedni red ili brojevnu vrijednost.

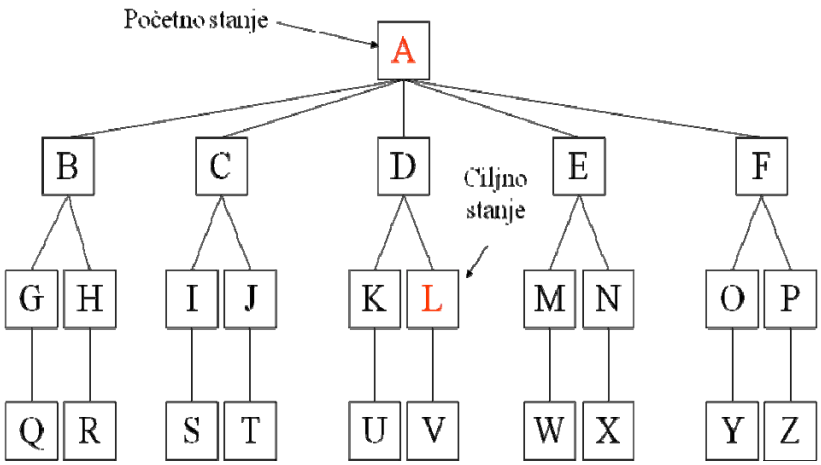


Slika 5. Poredak pretraživanja čvorova kod DFS

Pretraživanje započinje listom poredanih čvorova koje trebamo posjetiti. Čvor je struktura koja sadrži informacije o okolini ili dosegu problema (stanje ili vrijednost). Algoritam počinje postavljanjem početnog stanja problema (korijena stabla) na početak stoga. Prvo se posjeti početni čvor, a nakon toga se svi čvorovi povezani s njim dodaju na početak stoga. Ako pogledamo Sliku5: krećemo se od lijeva na desno - od tekućeg čvora (predstavljenog krugom u grafu) duž grana (predstavljenih poveznicama između čvorova) do čvorova povezanih s njim, dodavajući ih na stog. Nakon posjeta čvoru na početku stoga, njega se uklanja iz stoga. Pretraživanje se zatim premješta na sljedeći čvor u stogu i nastavlja se sve dok ne pronađemo što tražimo.

Algoritam pretraživanja po dubini:

1. Pohrani početni čvor u stog.
2. Ako je stog prazan onda vrati pogrešku i stani.
3. Ako je prvi element stoga ciljni čvor onda vrati nađeno uspješno rješenje i stani.
   Inače,
4. ukloni prvi čvor u stogu, razvij ga i svu njegovu djecu stavi na početak stoga bilo kojim redoslijedom.
5. Vrati se na korak 2.



Slika 6. Primjer grafa pretraživanja po dubini (modificirano prema [KEND2001])

Na Slici6. možemo vidjeti primjer grafa pretraživanja po dubini sa početnim stanjem A i ciljnim stanjem L. Dakle, početno stanje nam je čvor A te ga pohranjujemo u stog. U Tablici1 pratimo tijek izvođenja operacija te stanje na stogu. Kako prvi element stoga nije ciljni čvor, element A se briše iz stoga i na početak stoga se stavljaju svi njegovi susjedni čvorovi tj. djeca (B, C, D, E i F). Vraćamo se na korak 2. Prvi element stoga, čvor B, nije ciljni čvor, element B se briše iz stoga, a na početak stoga se stavljaju njegova djeca, čvorovi G i H. Promotrimo prvi element stoga, čvor G. On nije ciljni čvor, briše se iz stoga i na početak stoga se stavljaju njegova djeca, čvor Q. Čvor Q nije ciljni čvor, briše se iz stoga. Kako čvor Q nema djecu, vraćamo se unatrag na prvi neistraženi čvor, u ovom primjeru čvor H. Čvor H se briše iz stoga i na početak stoga se stavljaju njegova djeca, čvor R. Postupak se ponavlja dok prvi element stoga ne bude ciljni čvor.

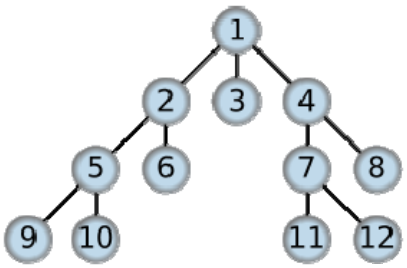
Tablica 1. Tijek izvođenja operacija

Operacija	Stanje na stogu
Pohrani A	A
Briše se A	B, C, D, E, F
Briše se B	G, H, C, D, E, F
Briše se G	Q, H, C, D, E, F
Briše se Q	H, C, D, E, F
Briše se H	R, C, D, E, F
Briše se R	C, D, E, F
Briše se C	I, J, D, E, F
Briše se I	S, J, D, E, F
Briše se S	J, D, E, F
Briše se J	T, D, E, F
Briše se T	D, E, F
Briše se D	K, L, E, F
Briše se K	U, L, E, F
Briše se U	L, E, F
L je ciljni čvor!	

Pretraživanje po dubini pogodnije je od pretraživanja po širini kada stablo pretraživanja ima puno ciljnih čvorova. Što se tiče potpunosti, ova strategija pretraživanja ne garantira pronalazak rješenja osim kada se radi o konačnim prostorima.

3.1.2. Pretraživanje po širini

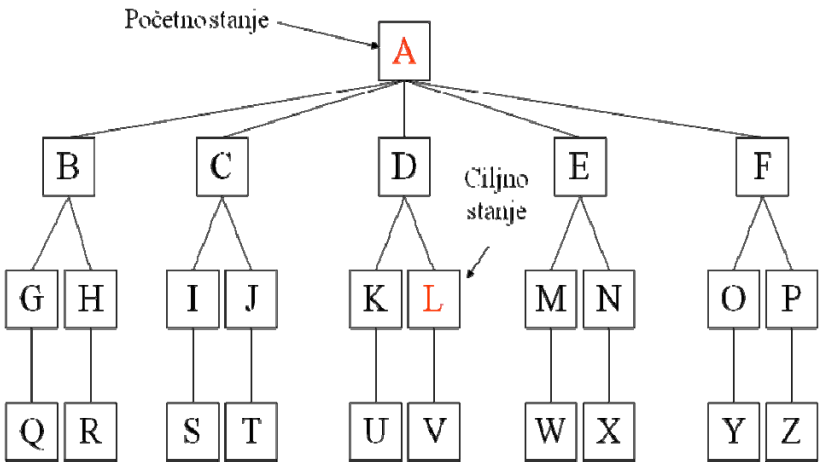
Pretraživanje po širini (eng. breadth first search, BFS) je način pretraživanja u kojem se koristi red kao struktura podataka u koju ćemo pohranjivati one vrhove grafa koje smo posjetili. Pretraživanje po širini jedna je od najčešće korištenih metoda pretraživanja. Ova metoda pretraživanja radi na principu FIFO (eng. first in first out), tj. prvi element koji je stavljen u red se i prvi briše. Element se stavlja na kraj liste, i element se uzima s početka liste. Algoritam za pretraživanje po širini nastoji se što više zadržati "blizu" početnog vrha. Algoritam počinje postavljanjem početnog stanja problema (korijena stabla) na početak reda. Prvo se posjeti početni čvor, a nakon toga se svi čvorovi povezani s njim dodaju u red. Ako pogledamo Sliku7 (stablo): krećemo se od lijeva na desno - od tekućeg čvora (predstavljenog krugom u grafu) duž grana (predstavljenih poveznicama između čvorova) do čvorova povezanih s njim, dodavajući ih u red.



Slika 7. Poredak pretraživanja čvorova kod BFS

Algoritam pretraživanja po širini:

1. Pohrani početni čvor u red.
2. Ako je red prazan onda vrati pogrešku i stani.
3. Ako je prvi element reda ciljni čvor onda vrati nađeno uspješno rješenje i stani.
   Inače,
4. ukloni prvi čvor u redu, razvij ga i svu njegovu djecu stavi na kraj reda bilo kojim redoslijedom.
5. Vrati se na korak 2.



Slika 8. Primjer grafa pretraživanja po širini (modificirano prema [KEND2001])

Na Slici8. možemo vidjeti primjer grafa pretraživanja po širini sa početnim stanjem A i ciljnim stanjem L. Dakle, početno stanje je čvor A te se pohranjuje u red. U Tablici2 pratimo tijek izvođenja operacija te stanje u redu. Kako prvi element reda nije ciljni čvor, element A se briše iz reda i na kraj reda stavljaju se svi njegovi susjedni čvorovi tj. djeca (čvorove B, C, D, E i F). Vraćamo se na korak 2. Prvi element reda čvor B nije ciljni čvor, briše se iz reda, a na kraj reda stavljaju se njegova djeca, čvorovi G i H. Promotrimo prvi element reda, čvor C. On nije ciljni čvor, briše se iz reda i na kraj reda stavljaju njegova djeca, čvorovi I i J. Čvor D nije ciljni čvor, briše se iz reda i na kraj reda stavljaju se njegova djeca, čvorovi K i L. Promotrimo stanje u redu: čvor E nije ciljni čvor, briše se iz reda i na kraj reda stavljaju se njegova djeca, čvorovi M i N. Postupak pretraživanja čvorova se ponavlja sve dok na početku reda ne bude čvor L, tj. ciljni čvor.

Tablica 2. Tijek izvođenja operacija

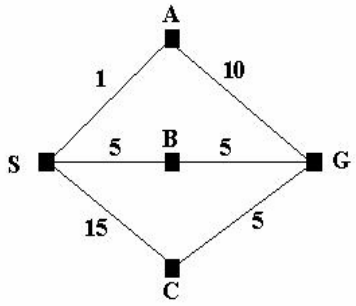
Operacija	Stanje u redu
Pohrani A	A
Briše se A	B, C, D, E, F
Briše se B	C, D, E, F, G, H
Briše se C	D, E, F, G, H, I, J
Briše se D	E, F, G, H, I, J, K, L
Briše se E	F, G, H, I, J, K, L, M, N
Briše se F	G, H, I, J, K, L, M, N, O, P
Briše se G	H, I, J, K, L, M, N, O, P, Q
Briše se H	I, J, K, L, M, N, O, P, Q, R
Briše se I	J, K, L, M, N, O, P, Q, R, S
Briše se J	K, L, M, N, O, P, Q, R, S, T
Briše se K	L, M, N, O, P, Q, R, S, T, U
L je ciljni čvor!	

3.1.3. Pretraživanje s jednolikom cijenom

Pretraživanje s jednolikom cijenom (eng. uniform cost search, UCS) je slično pretraživanju po širini. Razlika je u tome što se kod ove metode čvorovi koji su dodani u red sortiraju po cijeni

putanje (udaljenost od početnog čvora) i tek onda uklanjaju iz reda. Dakle, uklanja se uvijek onaj čvor koji ima najmanju cijenu putanje.

Pogledajmo sljedeći problem (Slika9). Želimo pronaći najkraći put od čvora S do čvora G, tj. čvor S nam je početni čvor, a čvor G cilj. Kao što možemo vidjeti S-B-G je najkraći put, a ako se primjenjuje metoda pretraživanja po širini onda će rješenje biti put S-A-G.



Slika 9. Primjer pretraživanja s jednolikom cijenom [KEND2001]

Pogledajmo kako će ova metoda pretraživanja doći do rješenja. Tijek izvođenja operacija možemo pretiti u Tablici3. Započinjemo sa početnim čvorom, čvor S, i pogledamo njegove susjedne čvorove (A, B i C). Čvor S se briše iz reda (jer nije ciljni čvor) te se u red dodaju njegova djeca, čvorovi A, B i C i to sljedećim redoslijedom: A(1), B(5) i C(15). Vrijednosti u zagradama predstavljaju cijenu putanje po kojoj se čvorovi sortiraju u redu. Najmanja cijena putanje je do čvora A pa se on promatra kao sljedeći čvor, kako nije ciljni čvor briše se iz reda i u red se dodaje njegovo dijete, čvor G. Promotrimo stanje u redu: B(5), G(11), C (15). Međutim algoritam ga ne prepoznaje kao konačno rješenje jer postoji još čvorova sa manjom cijenom putanje. Dakle, algoritam slaže u red sve cijene puta pojedinih čvorova i kao rješenje daje ono sa najmanjom vrijednošću. Čvor B se briše iz reda i u red se dodaje čvor G sa cijenom putanje 10. Sada je stanje u redu: G(10), G(11), C(15). Pronađeno je rješenje sa najkraćom cijenom putanje G(10), tj. put S-B-G.

Ovakvo pretraživanje će uvijek pronaći optimalno rješenje ako ono postoji.

Tablica 3. Tijek izvođenja operacija

Operacija	Stanje u redu
Pohrani S	S
Briše se S	A_1, B_5, C_{15}
Briše se A	B, G_{11}, C_{15}
Briše se B	G_{10}, G_{11}, C_{15}
G_{10} je ciljni čvor!	

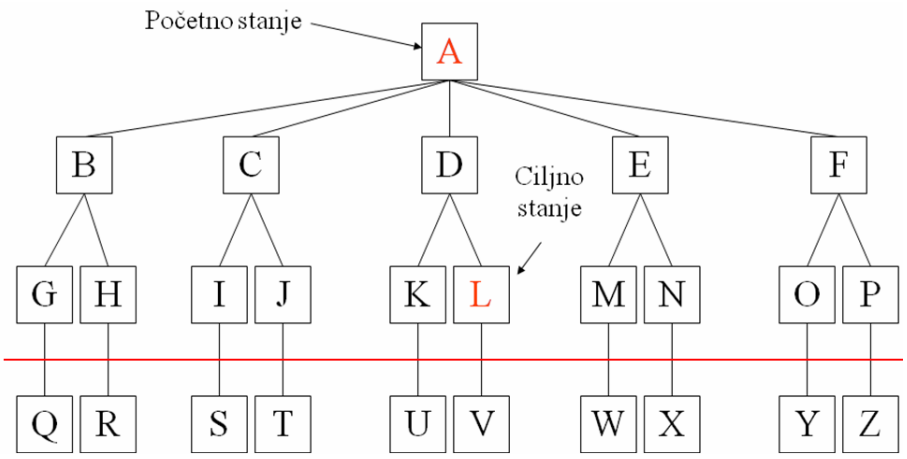
3.1.4. Pretraživanje do određene dubine

Pretraživanje do određene dubine (eng. depth limiting search, DLS) je vrsta pretraživanja po dubini. Tehnika pretraživanja je ista kao i kod pretraživanja po dubini, ali za razliku od tog

pretraživanja dubina pretraživanja je ograničena. Čak i ako se pretraživanje može nastaviti na sljedeće čvorove, unatoč ograničenoj dubini, ono se neće izvšiti i tako dubina pretraživanja neće ići u beskonaćno.

Algoritam pretraživanja do određene dubine:

1. Pohrani početni čvor u stog.
2. Ako je stog prazan onda vrati pogrešku i stani.
3. Ako je prvi element stoga ciljni čvor onda vrati nađeno uspješno rješenje i stani.
   Inače,
4. ako je čvor na dubini manjoj od d, razvij ga i svu njegovu djecu stavi na početak stoga bilo kojim redoslijedom.
5. Vrati se na korak 2.



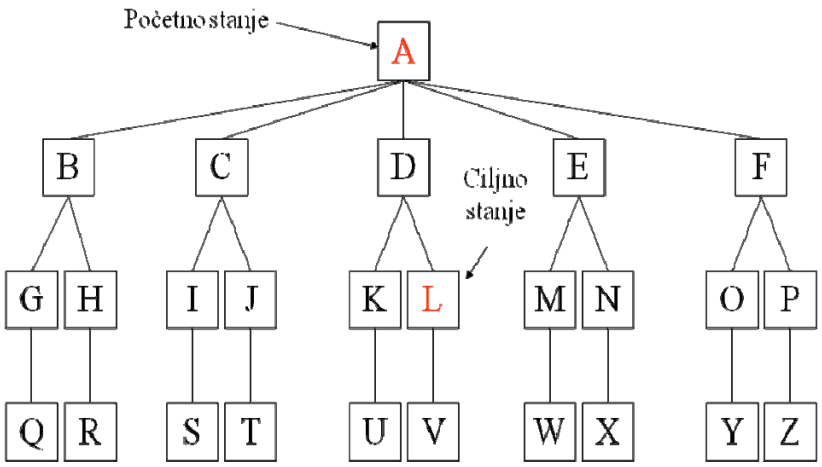
Slika10. Primjer pretraživanja do određene dubine (modificirano prema [KEND2001])

Na Slici10. možemo vidjeti primjer grafa pretraživanja po dubini sa početnim stanjem A i ciljnim stanjem L, dok je dubina pretraživanja d=2, tj. ne pretražuju se čvorovi na dubini većoj od d. Dakle, početno stanje nam je čvor A te ga pohranjujemo u stog. Kako prvi element stoga nije ciljni čvor, element A se briše iz stoga i na početak stoga se stavljaju svi njegovi susjedni čvorovi tj. djeca (B, C, D, E i F). Vraćamo se na korak 2. Prvi element stoga, čvor B, nije ciljni čvor, element B se briše iz stoga, a na početak stoga se stavljaju njegova djeca, čvorovi G i H. Promotrimo prvi element stoga, čvor G. On nije ciljni čvor, briše se iz stoga, ali se ne proširuje na susjedne čvorove budući da se ne nalazi na dubini manjoj od d=2. Pretraživanje se nastavlja na čvor H. Postupak se ponavlja dok prvi element stoga ne bude ciljni čvor.

3.1.5. Iterativno pretraživanje po dubini

Iterativno pretraživanje po dubini (eng. iterative deepening search, IDS) je vrsta pretraživanja koja kombinira svojstva pretraživanja po dubini i pretraživanja po širini. U tom pretraživanju se pretraživanje do određene dubine pokreće više puta, povećavajući granicu dubine sa svakom iteracijom dok se ne dosegne dubina d, tj. najkraća dubina do ciljnog stanja. Sa svakom iteracijom, ovo pretraživanje posjećuje čvorove istim redoslijedom kao i pretraživanje po dubini, ali redoslijed koji čvor će biti prvi posjećen je kao kod pretraživanja po širini.

Na primjeru grafa sa početnim stanjem A i ciljnim stanjem L vidjet ćemo kako metoda iterativnog pretraživanja po dubini dolazi do rješenja (Slika11). Ova metoda pretraživanja pretražuje sve moguće nivoe dubine: prvo 0, zatim 1, onda 2, itd. dok ne pronađe rješenje.

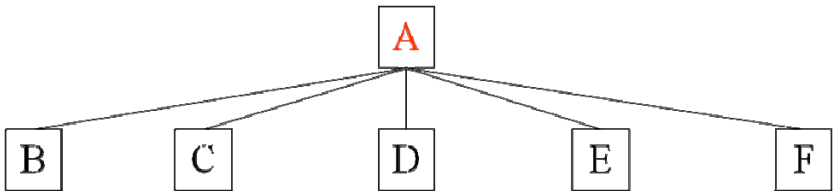


Slika 11. Primjer iterativnog pretraživanja po dubini (modificirano prema [KEND2001])

Nivo 0:

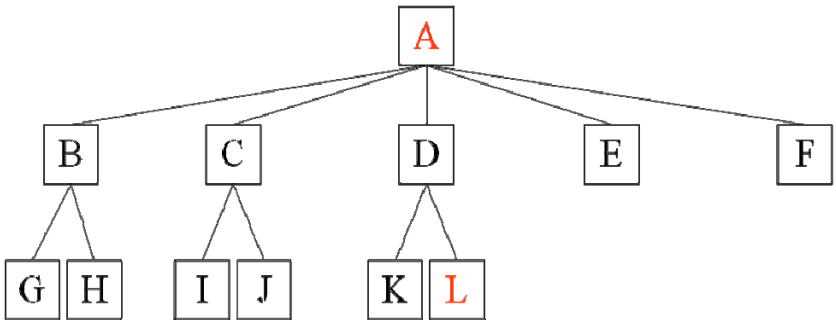
Pretraživanje započinjemo sa početnim stanjem čvorom A koji je dodan u red. Čvor A je potom proširen (eng. expanded) i izbrisan iz reda. Kako je ovo iteracija nultog reda (nivo dubine je 0) ne može se pretraživati nijedan nivo veći od nule. Time ova iteracija završava i slijedi iteracija rednog broja 1.

Nivo 1:



Opet započinjemo sa početnim stanjem, čvorom A. Čvor A je proširen, izbrisan iz reda i na početak reda su dodani otkriveni čvorovi (eng. revealed nodes), tj. njegova djeca, čvorovi B, C, D, E i F. Pretraživanje se nastavlja na nivou 1. Čvor B je proširen i uklonjen iz reda. Potom se pretražuje prvi neposjećeni čvor na nivou 1, tj. čvor C, proširuje se i briše iz reda. Pretraga se nastavlja dok svi čvorovi na prvom nivou ne budu posjećeni. Kako je ovo iteracija rednog broja 1 ne pretražuje se nijedan nivo većeg rednog broja. Time iteracija rednog broja 1 završava i prelazi se na iteraciju rednog broja 2.

Nivo 2:

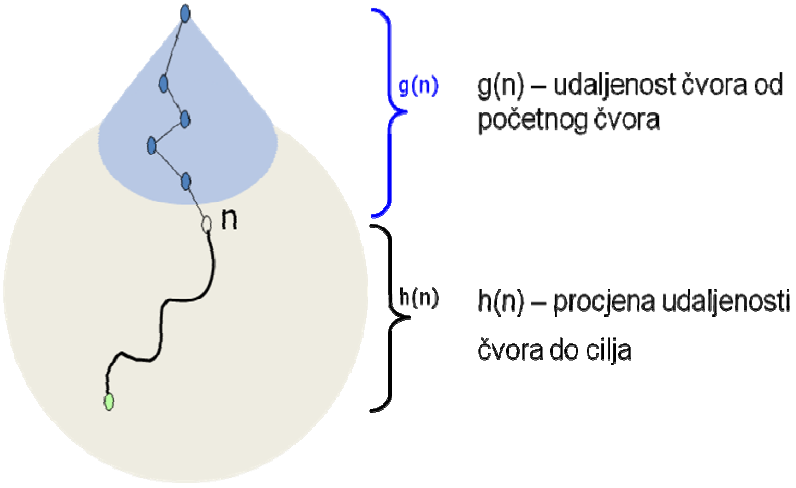


Pretraživanje opet započinje sa početnim stanjem, čvorom A. Čvor A uklanja se iz reda, čvorovi B, C, D, E i F su dodani na početak reda (nivo 1). Prelazi se na pretraživanje nivoa 2. Čvor B je posjećen, briše se iz reda i čvorovi G i H su dodani na početak reda. Čvor G je proširen i uklonjen iz reda. Potom se pretražuje prvi neposjećeni čvor na drugom nivou, čvor H, proširuje se i briše iz reda. Potom se vraćamo na posjećeni čvor C, briše se iz reda i na početak reda se dodaju čvorovi I i J. Postupak se ponavlja do ciljnog stanja, čvora L.

3.2. USMJERENE METODE PRETRAŽIVANJA

Za razliku od slijepih metoda pretraživanja, usmjerene ili heurističke metode pretraživanja koriste dodatne informacije o problemu koje mogu bitno ubrzati pronalaženje rješenja, što je više podataka dostupno metodi pretraživanja to je pretraživanje efikasnije. Drugim riječima, usmjerene metode pretraživanja koriste dodatne informacije kako bi pronašle što kraći put od početnog stanja do ciljnog stanja. Te dodatne informacije su npr. informacije o naravi stanja, o

cijeni prijelaza iz jednog stanja u drugo, o osobinama cilja, itd. Takve informacije obično se nazivaju *heurističke* - temelje se na iskustvenim pravilima i tehnikama prosuđivanja koje mogu pomoći, ali nužno ne osiguravaju pronalaženje rješenja. Heurističke informacije mogu se oblikovati u heurističku evaluacijsku funkciju koja zavisi od pojedinog čvora n i od cilja koji se traži, f(n) = g(n) + h(n) (Slika12). Dakle, ukupna vrijednost heurističke funkcije f(n) dobiva se zbrajanjem vrijednost h(n) i g(n). Pri tome je h(n) funkcija procjene udaljenosti čvora n od ciljnog čvora, a g(n) udaljenost čvora n od početnog čvora [PAVK2005].



Slika 12. Heuristička funkcija (modificirano prema [GOFO2003])

Vrijednost g(n) se koristi samo u slučaju A* pretraživanja. Ovi algoritmi ne pretražuju čvorove po redu nego za svaki obiđeni čvor izračunavaju vrijednost heurističke funkcije te se pretraživanje usmjerava na čvor sa najmanjom vrijednošću heurističke funkcije. Većina algoritama za usmjereno pretraživanje namijenjena je za strukture stabla. Jedan od nedostataka usmjerenih metoda pretraživanja je izravna ovisnost o odabiru heurističke funkcije. Ukoliko je ona dobro odabrana, usmjerene metode će neusporedivo brže doći do rješenja nego slijepe metode. Međutim, pogrešnim odabirom heurističke funkcije pretraživanje će biti vrlo neučinkovito.

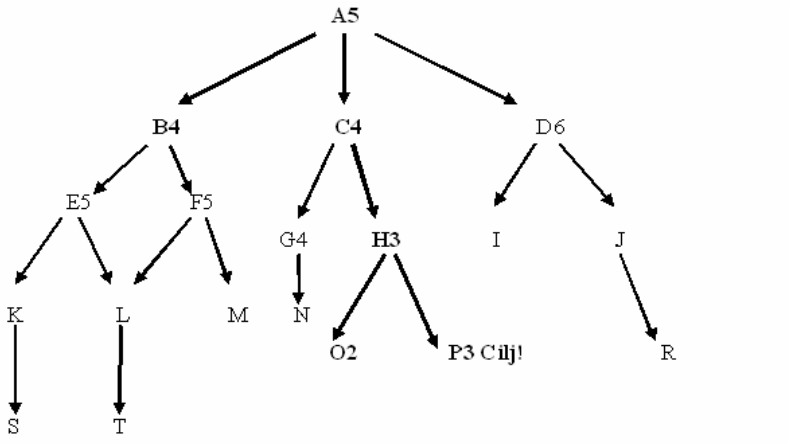
3.2.1. Pretraživanje najboljim prvim

Pretraživanje najboljim prvim (eng. best first search) je usmjereno pretraživanje koje optimizira (poboljšava) pretraživanje po širini. Funkcijom heuristike se procjenjuju čvorovi u grafu i odabire se najbolje procijenjeni čvor za daljnje pretraživanje. Ovo pretraživanje je opisano kao procjenjivanje obećavajućeg čvora n pomoću heurističke evaluacijske funkcije f(n) koja općenito ovisi o opisu čvora n, opisu cilja, podacima prikupljenim tijekom pretraživanja sve do tog trenutka i o dodatnom znanju o domeni problema.

Algoritam pretraživanja najboljim prvim:

1. Pohrani početni čvor u red.
2. Ako je red prazan onda vrati pogrešku i stani.
3. Ako je prvi element reda ciljni čvor onda vrati nađeno uspješno rješenje i stani.
   Inače,
4. Ukloni prvi čvor u redu, razvij ga i izračunaj procjenu udaljenosti od cilja za svako dijete. Stavi svu djecu u red i uredi sve elemente reda po procjeni udaljenosti od cilja.
5. Vrati se na korak 2.

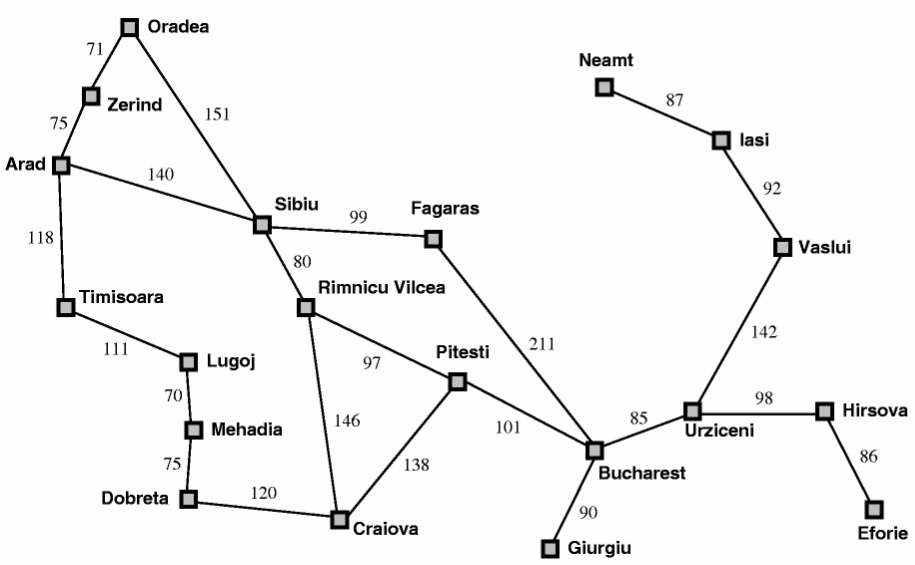
Na primjeru grafa sa početnim stanjem A i završnim stanjem P vidjet ćemo kako metoda pretraživanja najboljim prvim dolazi do rješenja (Slika13). Brojevi pridodani čvorovima predstavljaju udaljenost pojedinog čvora od cilja.



Slika 13. Primjer pretraživanja najboljim prvim [DALB2002]

Pretraživanje započinje sa početnim čvorom A5 koji se pohranjuje u red. Kako čvor A nije ciljni čvor briše se iz reda i sva njegova djeca se dodaju u red i to redoslijedom B4, C4, D6. Kako prvi element reda, čvor B4, nije ciljni čvor, briše se iz reda i u red se dodaju njegova djeca, čvorovi E5 i F5. Stanje u redu je: C4, E5, F5, D6. Prvi čvor u redu nije ciljni čvor, briše se iz reda i u red se dodaju njegova djeca, G4 i H3. Stanje u redu je: H3, G4, E5, F5, D6. Čvor H3 nije ciljni čvor, briše se iz reda i u red se dodaju njegova djeca, čvorovi O2 i P3. Stanje u redu je: O2, P3, G4, E5, F5, D6. Čvor O2 se briše iz reda. Stanje u redu je : P3, G4, E5, F5, D6. Prvi element reda je ciljni čvor i nađeno je rješenje.

Na sljedećem primjeru putovanja po Rumunjskoj demonstrirat ćemo pretraživanje najboljim prvim. Primjer je iz knjige ″Artificial Intelligence: A Modern Approach″, Russel i Norvig, 1995. Na karti je dana povezanost glavnog grada Bukurešta sa ostalim gradovima Rumunjske, a u tablici pravocrtna udaljenost pojedinih gradova od Bukurešta. Zadatak je pronaći najkraći put od grada Arad (početno stanje) do Bukurešta (ciljno stanje).

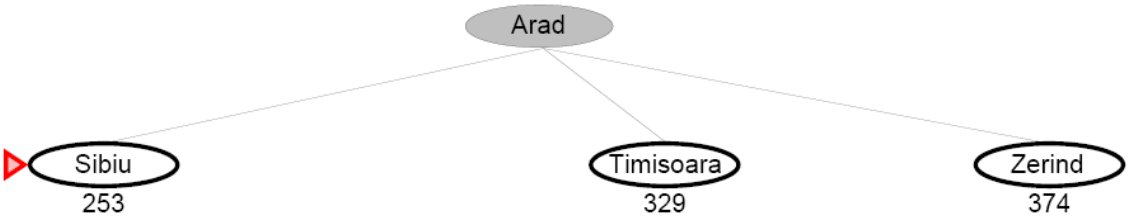


Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

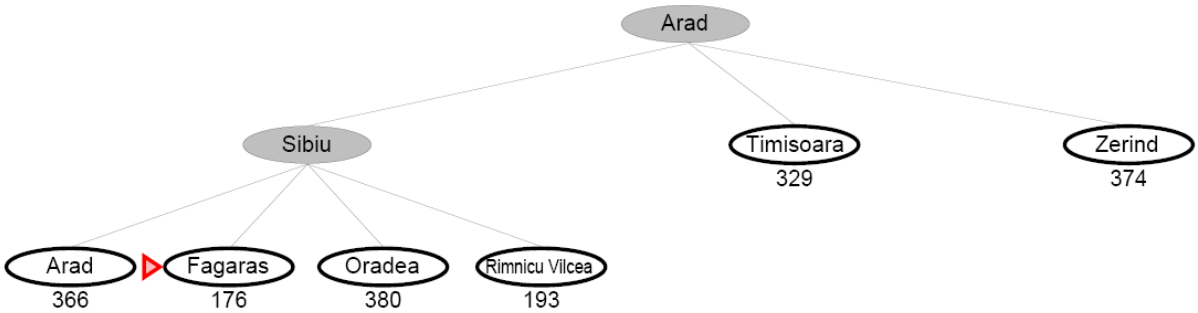
Slika 14. Primjer putovanja po Rumunjskoj

Pretraživanje započinje sa gradom Arad čija je prevocrtna udaljenost od Bukurešta 366.

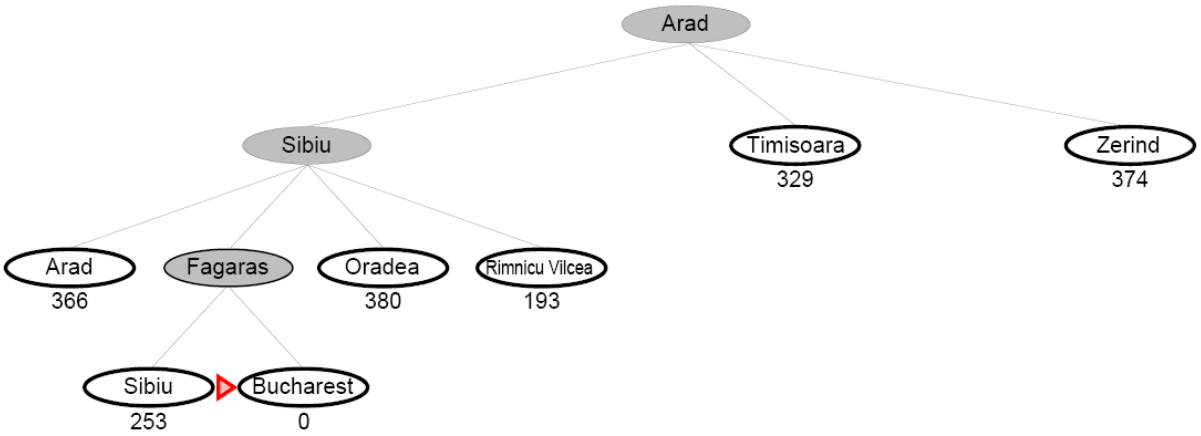
Proširujemo čvor Arad na čvorove Zerind, Sibiu i Timisoara. Njihova pravocrtna udaljenost od cilja je 374, 253 i 329 redom:



Kako grad Sibiu ima najmanju udaljenost od cilja on se sljedeći proširuje na čvorove, Arad, Fagaras, Oradea i Rimnicu čije su udaljenosti od cilja 366, 178, 380 i 193 redom:



Sljedeći koji se proširuje je Fagaras, obzirom da ima najkraću udaljenost od cilja, i to nas vodi do čvorova Sibiu i Bukurešta čija je udaljenost jednaka 0 i stigli smo do cilja:



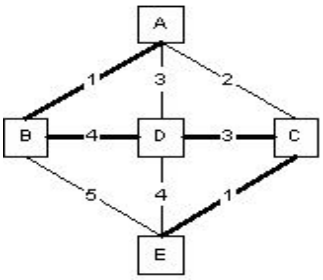
U slučaju pretraživanja najboljim prvim uvijek se odabire onaj čvor koji se čini najbliži cilju, no nađeno rješenje nije optimalno. Naime, putanja Sibiu-Fagaras-Bukurešt je za 32 milje duža nego putanja Sibiu-Rimnicu-Pitesti.

3.2.2. Pretraživanje penjanjem

Pretraživanje penjanjem ili metoda uspona na vrh (eng. hill climbing search) je slična metodi pretraživanja po dubini s tim da se širi onaj čvor koji je najpogodniji prema vrijednosti heurističke funkcije, dok se sve informacije o ostalim čvorovima brišu. Npr. heuristička funkcija može biti neka mjera udaljenosti od cilja, tada je najpogodniji čvor sa minimalnom vrijednošću. U primjeni se uglavnom traže minimalne vrijednosti heurističke funkcije dok se zemljopisne analogije odnose na maksimalne vrijednosti.

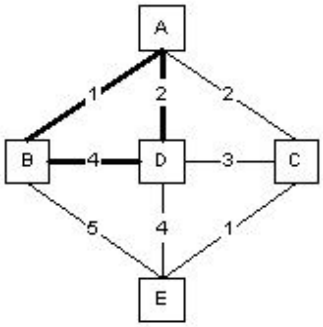
Algoritam pretraživanja penjanjem:

1. Pohrani početni čvor u red
2. Ako je red prazan onda vrati pogrešku i stani
3. Ako je prvi element reda ciljni čvor onda vrati nađeno uspješno rješenje i stani
   Inače,
4. Ukloni prvi element u redu, razvij ga i svu njegovu djecu dodaj na početak reda i to po procjeni udaljenosti čvora do cilja.
5. Vrati se na korak 2.



Slika 15. Primjer pretraživanja penjanjem

Slika15 ilustrira pretraživanje penjanjem počevši od početnog čvora A do ciljnog čvora E. Prvi sljedeći čvor koji se proširuje je čvor B, budući da ima manju vrijednost heurističke funkcije od čvorova C i D. Zatim se pretražuje čvor D, potom čvor C i pretraživanje završava dolaskom do ciljnog čvora E. Dakle, rješenje je pronađeno, međutim ono nije optimalno. Optimalno rješenje je putanja A-C-E. Nedostatak ove metode pretraživanja je u tome da ne održava stanja o svim čvorovima, za pretraživanje je važno samo trenutno stanje. Slika16 ilustrira pretraživanje penjanjem koje se zaglavi u petlji. Početno stanje je čvor A, a ciljno stanje čvor E. Pretraživanje se vrti u petlji A-B-D-A.



Slika 16. Primjer pretraživanja penjanjem

3.2.3. A pretraživanje*

A* pretraživanje je poseban slučaj pretraživanja najboljim prvim koji uključuje postupak računanja udaljenosti čvora n od početnog čvora. Za svaki čvor, koji je na putu od početnog do ciljnog čvora, algoritam stvara sve slijedne čvorove (djecu) i računa procjenu udaljenosti od početnog do ciljnog čvora kroz sve stvorene čvorove. Izabire onaj slijedni čvor za koji je ta procjena udaljenosti najmanja. Tada se stvaraju djeca odabranog čvora, računaju se procjene udaljenosti za putove kroz njih i postupak se nastavlja dok se ne pronađe rješenje.

Kao ukupna vrijednost heurističke funkcije nekog čvora obično se uzima izraz $f(n) = g(n) + h(n)$. Jasna je uloga funkcije h(n) u izračunavanju ukupne heurističke procjene udaljenosti od rješenja. Što je ta vrijednost manja, to smo bliže cilju. Ono što bi moglo zbunjivati je uloga funkcije g(n) koja predstavlja udaljenost čvora od početka. Zašto bi algoritmu bilo bitno koliko je daleko od početnog čvora kad mu je cilj pronaći ciljni. Već je rečeno da je osim pronalaženja ciljnog čvora

zadatak metoda pretraživanja i pronalaženje puta od početnog do ciljnog čvora. Zahtijeva se da je taj put što je moguće kraći. Funkcija g(n) služi upravo tome. Ukoliko je procijenjeno da su dva čvora jednako daleko od ciljnog čvora, algoritam će odabrati onaj koji je bliži početnom čvoru i na taj način pokušati u što manje pokušaja doći do rješenja, te pronaći kraći put.

Algoritam A* koristi dvije liste:

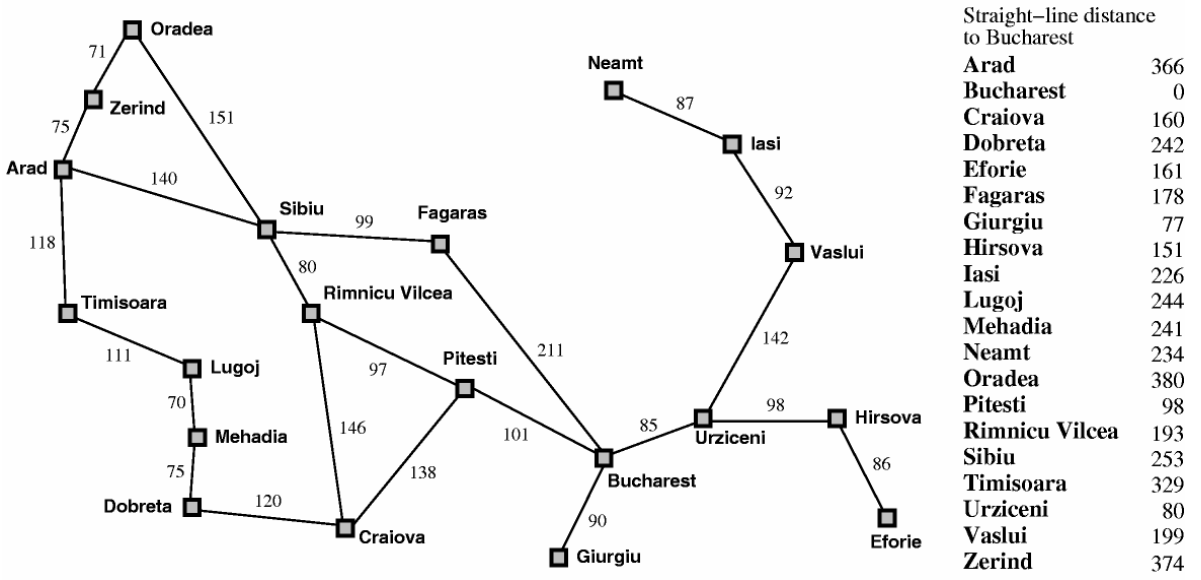
- OPEN – čvorovi koji su stvoreni, ali nisu još proširen
- CLOSED – čvorovi koji su proširen i čija djeca su na raspolaganju algoritmu pretraživanja

Lista OPEN složena je po rastućoj vrijednosti funkcije f. Proširuje se čvor koji je na vrhu liste (najmanji f), a prošireni čvor se stavlja na listu CLOSED. Djeca se stavljaju na listu OPEN koja se ponovo složi po rastućoj vrijednosti funkcije f.

Algoritam A*:

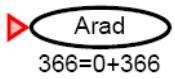
1. Stavi početni čvor na listu OPEN.
2. Ako je lista OPEN prazna onda se zaustavi i vrati pogrešku.
3. Ukloni sa liste OPEN čvor n koji ima najmanju vrijednost f(n). Ako je čvor ciljni čvor vrati uspješan završetak.
   Inače,
4. Proširi čvor n, stvarajući svu njegovu djecu. Za svako dijete n' izračunaj vrijednost f(n'), a čvor n stavi na listu CLOSED.
   Za svako dijete n', koje nije na listi OPEN ili CLOSED učini:
   - pridijeli mu izračunatu vrijednost f(n');
   - pridijeli mu pokazivač unazad (eng. backpointer) na čvor n;
   - stavi ga na listu OPEN;
5. Za svako dijete n' koje je na listi OPEN ili CLOSED učini:
   - pridijeli mu manju od vrijednosti f(n') izračunatu u ovom koraku ili prije;
   - ako je n' bio na listi CLOSED i vrijednost f(n') mu je promijenjena (smanjena), ukloni ga i stavi ga na listu OPEN te ažuriraj njegov pokazivač unatrag.
6. Vrati se na korak 2.

Na primjeru putovanja po Rumunjskoj demonstrirat ćemo pretraživanje A*. Zadatak je pronaći najkraći put od grada Arad (početno stanje) do Bukurešta (ciljno stanje).

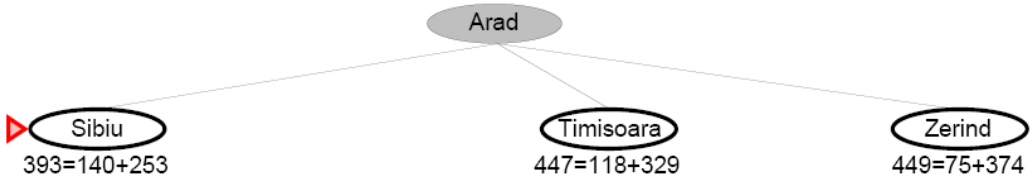


Slika 17. Primjer putovanja po Rumunjskoj

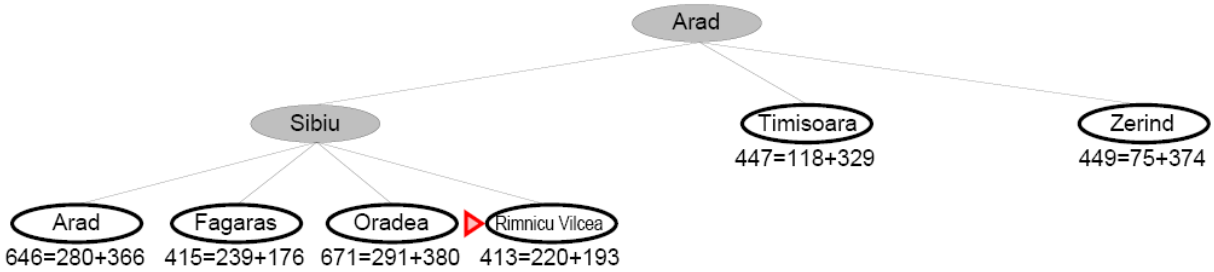
U ovom slučaju vrijednost g(n) predstavlja udaljenost pojedinih gradova od početnog čvora tj. grada Arad, a vrijednost h(n) udaljenost pojedinih gradova do ciljnog čvora, tj. grada Bukurešta. Pretraživanje započinjemo sa početnim čvorom, gradom Aradom. Njegova ukupna vrijednost heurističke funkcije iznosi f(n) = 0 + 366.



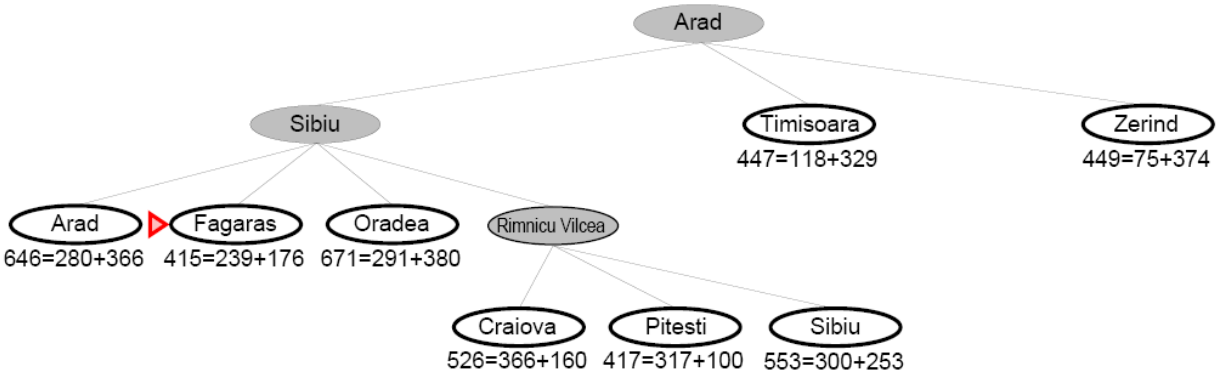
Proširujemo čvor Arad na njegove susjedne čvorove i za svaki čvor se izračunava vrijednost heurističke funkcije:



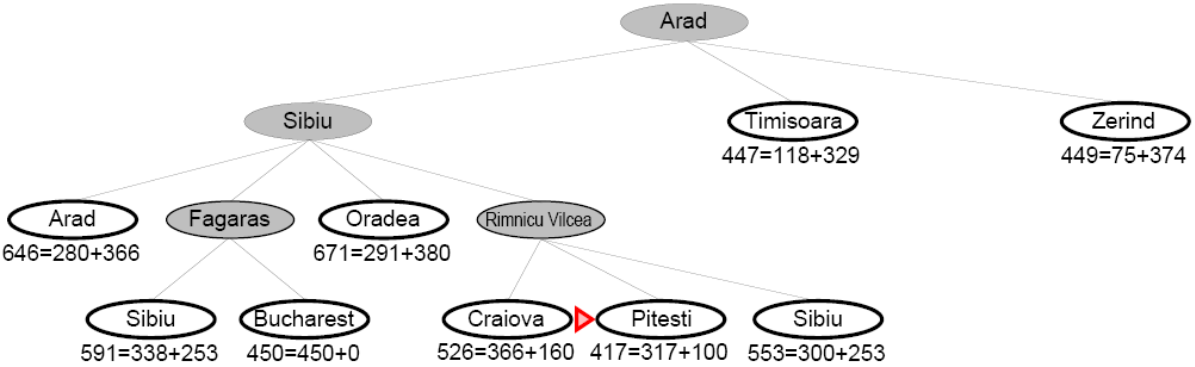
Proširuje se čvor koji ima najmanju vrijednost heurističke funkcije, u ovom slučaju čvor Sibiu i za svaki njegov susjedni čvor računa se vrijednost heurističke funkcije:



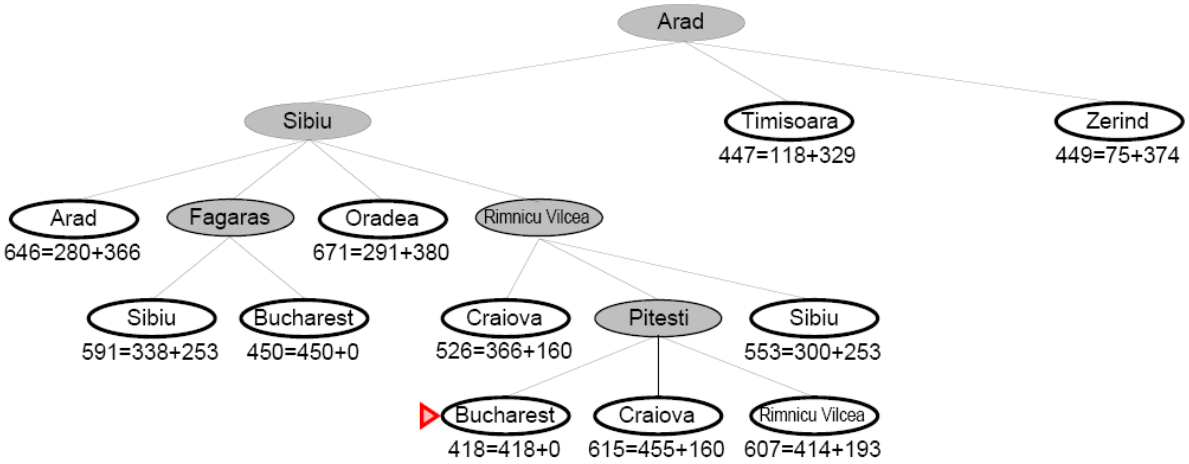
Potom se proširuje sljedeći čvor sa najmanjom vrijednosti heurističke funkcije, čvor Rimnicu:



Sljedeći čvor koji se proširuje je čvor Fagaras (od svih čvorova ima najmanju vrijednost heurističke funkcije):



Potom se proširuje čvor Pitesti i time smo došli do čvora Bukurešt, tj ciljnog čvora:

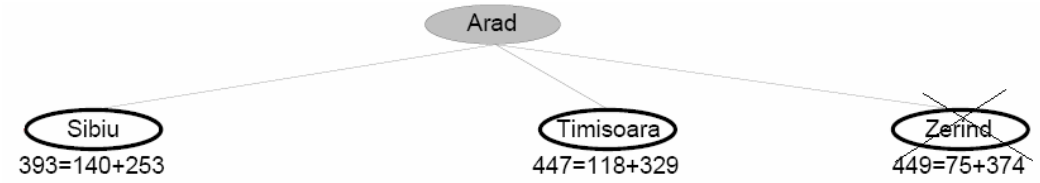


3.2.4. Ograničeno pretraživanje po širini

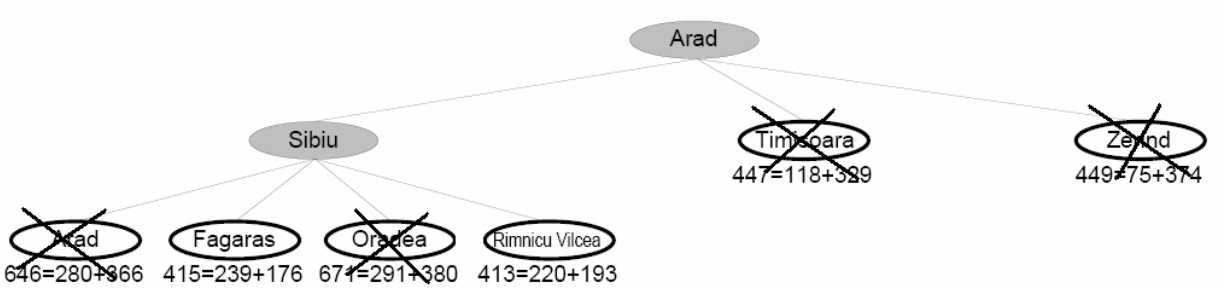
Ograničeno pretraživanje po širini ili pretraživanje snopom zraka (eng. beam search) je posebna vrsta A* pretraživanja. Ova metoda provodi pretraživanje na samo n odabranih najboljih čvorova. Pretraživanje započinje sa početnim stanjem koje se proširuje na susjedne čvorove. Ukoliko ni jedan od slijednih čvorova nije ciljni čvor, algoritam pretraživanja od svih čvorova odabire n najboljih čvorova za daljnje pretraživanje i postupak se ponavlja. Dakle, na svakoj razini odabiru se najbolji čvorovi (najbolji u smislu vrijednosti heurističke funkcije).

Na prethodnom primjeru ilustrirat ćemo tehniku ograničenog pretraživanja po širini u slučaju kada je n=2, tj. na svakoj razini se odabiru po dva najbolja čvora za daljnje pretraživanje. Pretraživanje započinjemo sa početnim čvorom, gradom Arad:

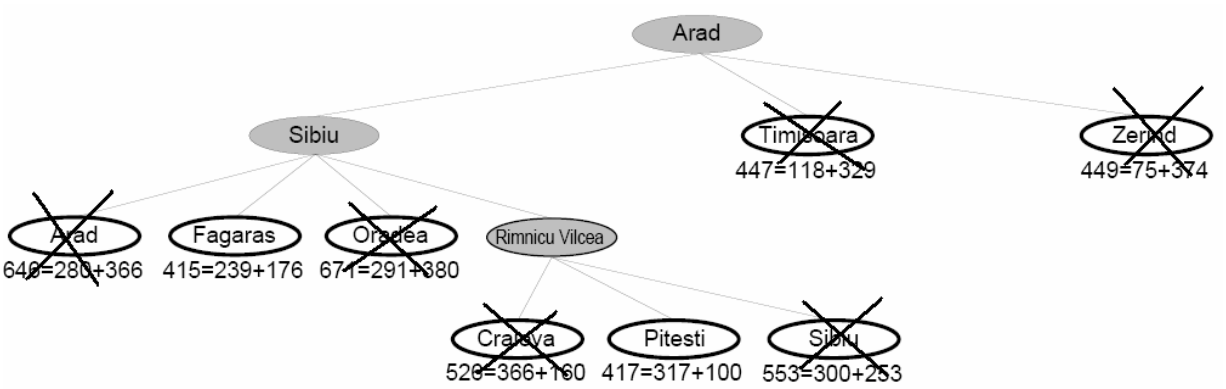
Kako to nije ciljni čvor, proširuje se na susjedne čvorove te se za daljnje pretraživanje biraju samo dva čvora sa najmanjim vrijednostima heurističke funkcije, čvorovi Sibiu i Timisoara:



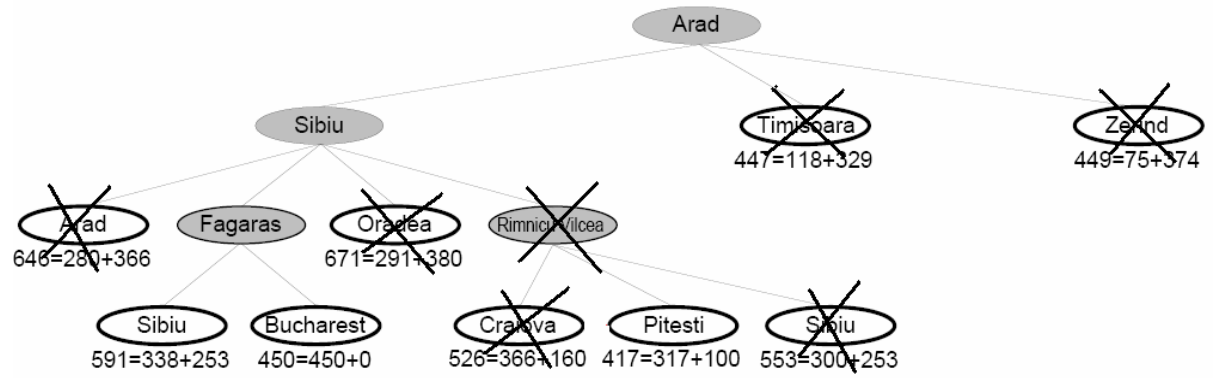
Čvor Sibiu se proširuje na slijedne čvorove (jer ima manju vrijednost heurističke funkcije od čvora Timisoara), te se od svih čvorova odabiru dva najbolja za daljnje pretraživanje, čvorovi Fagaras i Rimnicu:



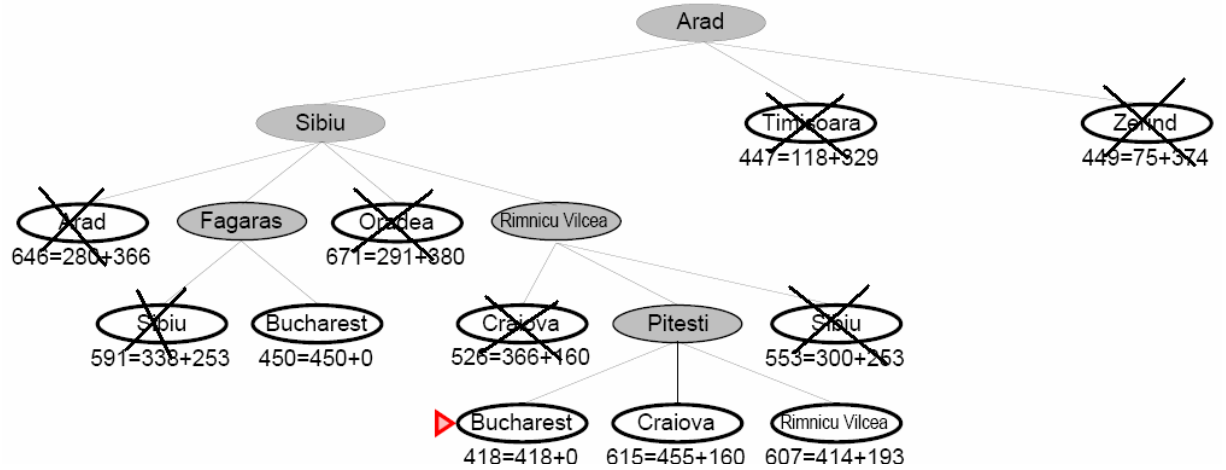
Kako čvor Rimnicu ima manju vrijednost heurističke funkcije od čvora Fagaras, on se dalje proširuje na čvorove Craiova, Pitesti i Sibiu. Od svih čvorova odabiru se dva najbolja te se pretraživanje nastavlja. Dakle, odabrani su čvorovi Fagaras i Pitesti:



Čvor Fagaras ima manju vrijednost heurističke funkcije od čvora Pitesti pa se on sljedeći proširuje na čvorove Sibiu i Bukurešt:



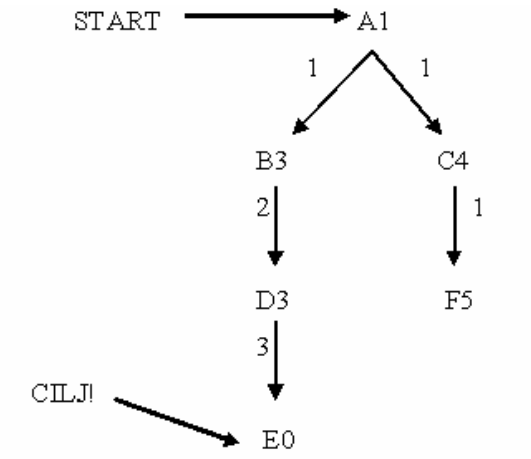
Odabiru se dva čvora sa najmanjom vrijednosti heurističke, čvorovi Bukurešt i Pitesti, te se proširuje onaj sa najmanjom vrijednosti heurističke funkcije. Dakle, čvor pitesti se proširuje na svoje susjedne čvorove i od svih čvorova čvor sa najmanjom vrijednosti heurističke funkcije je ujedno i ciljni čvor:



3.2.5. IDA pretraživanje*

Iterativno A* pretraživanje po dubini (eng. Iterative deeping A* search, IDA*) je posebna vrsta A* pretraživanja. Ova metoda pretraživanja postavlja graničnu vrijednost (eng. threshold) tj. dubinu (uglavnom se uzima vrijednost heurističke funkcije početnog čvora) do koje se pretražuje. Ako rješenje nije pronađeno na postavljenoj dubini, dubina pretraživanja se povećava tako da se uzima najmanja vrijednost heurističke funkcije neproširenih čvorova i pretraživanje se ponavlja.

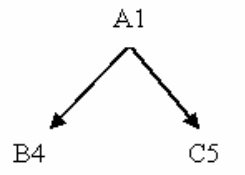
Na primjeru grafa sa početnim stanjem A i završnim stanjem E ilustrirat ćemo IDA* pretraživanje (Slika18). Brojevi pridodani čvorovima predstavljaju udaljenost čvora od cilja, a brojevi između čvorova predstavljaju udaljenost među pojedinim čvorovima.



Slika 18. Primjer IDA* pretaživanja

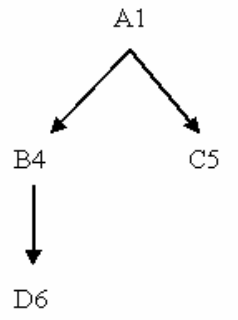
Pretraživanje započinjemo od početnog čvora A i postavljamo njegovu ukupnu vrijednost heurističke funkcije f(n) = 0 + 1 = 1 kao graničnu vrijednost. Kako čvor A nije ciljni čvor, proširujemo ga na čvorove B i C računamo njihovu vrijednost heurističke funkcije:

granična vrijednost = 1



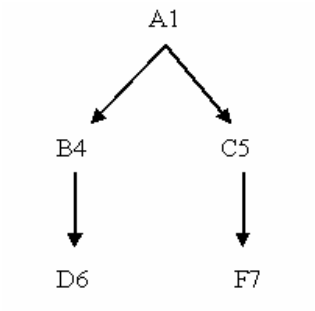
Kako cilj na dubini 1 nije pronađen pretraživanje se ponavlja, a za graničnu vrijednost se uzima najmanja vrijednost heurističke funkcije na prethodnoj dubini. Dakle, granična vrijednost je sada jednaka 4, te se čvor B obzirom da nije ciljni čvor proširuje na čvor D:

granična vrijednost = 4



Dakle, ni na ovoj dubini cilj nije pronađen, pretraživanje se ponavlja, granična vrijednost uvećava, a čvor C se proširuje na čvor F.

granična vrijednost = 5



Na ovoj dubini cilj također nije pronađen pa se pretraživanje opet ponavlja, granična vrijednost jednaka je 6, čvor D se proširuje na čvor E koji je ujedno i ciljni čvor. Dakle, putanja koja vodi do rješenja je A-B-D-E.

4. PRIMJERI INTELIGENTNIH AGENATA ZA PRETRAŽIVANJE WEB-A

Do danas je razvijen velik broj različitih inteligentnih agenata koji nam pomažu pri pretraživanju Web-a. Njihov razvoj započeo je još ranih 90-tih i još uvijek traje. Popis svih inteligentnih agenata može se vidjeti u bazi podataka, na adresi http://www.robotstxt.org/db.html. U ovom poglavlju dat ćemo primjere nekih od njih. Dan je kratki pregled njihovih funkcija i osnovnih svojstava. Obzirom na godine nastanka pojedinih agenata, stranice za pristup nisu više u funkciji pa stoga nismo bili u mogućnosti da ih pokrenemo, dok korištenje drugih nije besplatno.

4.1. TueMosaic

Najraniji primjer inteligentnog agenta za pretraživanje Web-a je *tueMosaic* [CHAU2003]. Koristeći tueMosaic, korisnik može unijeti ključnu riječ, odrediti dubinu i širinu pretraživanja za linkove sadržane na početnoj stranici, te zatražiti od agenta da hvata (eng. fetch) početne stranice povezane sa trenutnom početnom stranicom. Metoda pretraživanja koju koristi agent tueMosaic je pretraživanje najboljim prvim.

4.2. WebCrawler

Nakon njega razvijen je inteligentni agent *WebCrawler*, koji proširuje koncept tueMosaic agenta time što započinje pretraživanje vlastitog indeksa, a linkove pretražuje na inteligentan način [YANG2000]. On stvara indeks koristeći metodu pretraživanja po širini, time omogućuje svakom poslužitelju (eng. server) sa korisnim sadržajima nekoliko stranica zastupljenih u indeksu. Indeks agenta WebCrawler sastoji se od gotovo 50 000 dokumenata distribuiranih na 9 000 različitih poslužitelja, svakodnevno odgovarajući na oko 6 000 upita. Nakon preuzimanja dokumenta, WebCrawler obavlja 3 akcije: označava dokument kao dohvaćenog (eng. retrieved), dešifrira (eng. deciphers) odlazeće linkove (eng. outbound link) i indeksira sadržaj dokumenta. Svi ovi koraci uključuju i spremanje podataka u bazu podataka. WebCrawler pretraživanje započinje od poznatog skupa dokumenata, istražujući njihove odlazeće linkove, slijedeći jednog od linkova koji vodi u novi dokument, a zatim ponavlja cijeli postupak. Koji link će slijediti određuje na temelju sličnosti teksta koji opisuje link (eng. anchor text) sa korisnikovim upitom. Iako

pretraživanje na ovaj način dobro radi, ti tekstovi su uglavnom kratki i ne osiguravaju relevantnost podataka. Dakle, korisnik unese ključne riječi kao upit, a naslovi i URL dokumenata koji sadrže neke ili sve od tih riječi su dobavljeni iz indeksa, i predstavljeni korisniku kao lista dokumenata sortiranih po važnosti [PINK1994]. Na Slici19 možemo vidjeti izgled WebCrawler zaslona. Pristup je omogućen klikom na link http://www.webcrawler.com .

Slika19. Izgled WebCrawler zaslona

4.3. TkWWW

TkWWW agent je program integriran u TkWWW pretraživaču [CHAU2003]. Može biti odaslan (eng. dispatched) iz pretraživača i pretraživati Web susjedstvo kako bi pronašao relevantne stranice i vratio listu linkova koji ukazuju na odgovarajuće dokumente. Agent pretražuje Web u skladu sa funkcijama opisanih pomoću Tcl nastavka (eng. extension) razvijenih za TkWWW pretraživač. Također može biti korišten za izradu HTML indeksa, izradu WWW statistika, sakupljanje mapa ili slika, te izvođenje bilo koje druge funkcije opisane pomoću TkWWW Tcl nastavka. Pretraživanja mogu biti ograničena s obzirom na URL imena, granicu na duljini putanje, itd. Glavna prednost TkWWW agenta je njegova fleksibilnost u prilagođavanju promjenama u okolini [SPET1994].

4.4. WebAnts

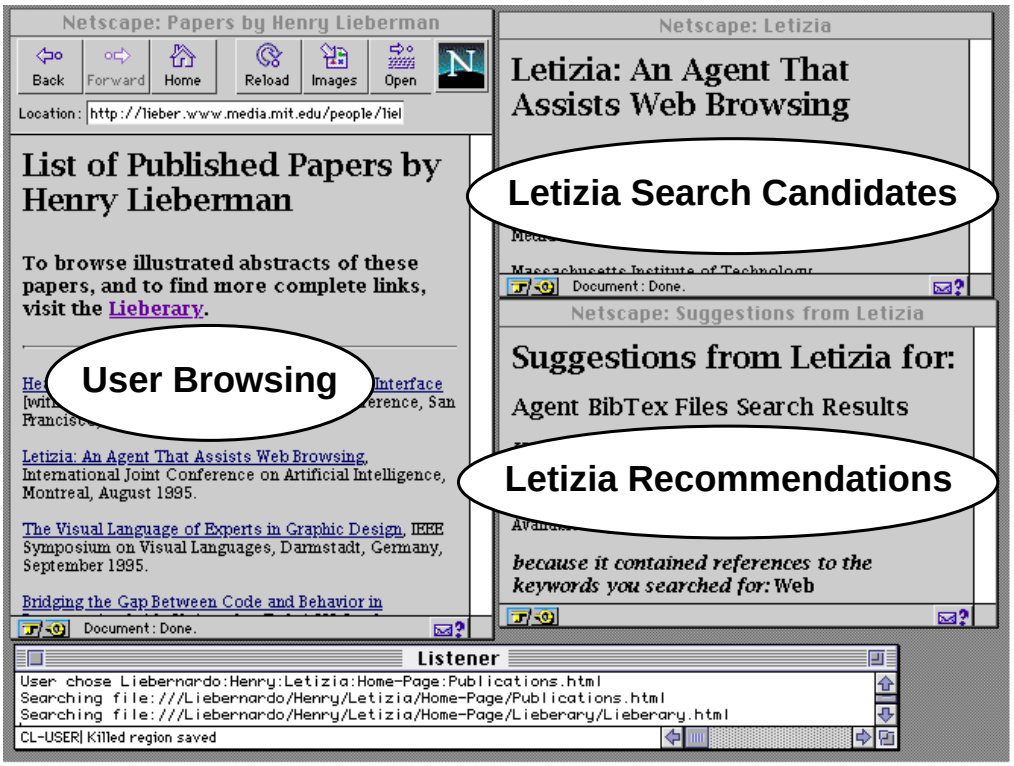
WebAnts projekt sastoji se od razvijanja agenata za pretraživanje ili mravi (eng. ants) koji međusobno surađuju tako da dijele rezultate pretraživanja. Kad jedan agent pronađe dokument koji zadovoljava kriterij pretraživanja, rezultate pretraživanja dijeli sa ostalim agentima tako da oni ne pretražuju isti dokument, te ako jedan agent pretražuje neki dokument on obavještava druge agente koji dokument pretražuje tako da oni pretraživanje usmjere na druge dokumente [YANG2000].

4.5. RBSE

RBSE (eng. Repository Based Software Engineering) agent je bio prvi agent koji je indeksirao dokumente po sadržaju. Dokumente sa Web-a dobavlja na način da sakuplja URL-ove koje pohranjuje u bazu podataka. To čini koristeći Mite program, koji uzima URL kao parametar i vraća stranicu ili opravdanje za pogrešku. Pri tome koristi nekoliko tehnika pretraživanja: pretraživanje po širini i ograničeno pretraživanje po dubini [YANG2000].

4.6. Letizia

Letizia je primjer inteligentnog agenta koji pomaže korisniku pri pretraživanju Web-a [LIEB1995]. Dok korisnik koristi Web pretraživač kao npr. Netscape, agent prati korisnikovo ponašanje i pokušava otkriti njegovo područje interesa. Ovaj model pretraživanja informacija sastoji se od međusobne suradnje korisnika i agenta. Letizia koristi Netscape kao svoje sučelje, sa jednim prozorom namijenjenim pretraživanju korisnika te jednim ili više prozora koji prikazuju preporuke od agenta. Letizia i korisnik istodobno pregledavaju isti prostor pretraživanja pokušavajući pronaći dokumente od interesa. Sukladno sa pretraživanjem korisnika, Letizia pokušava predvidjeti buduće potrebe korisnika. Korisnik se ne ometa tijekom pretraživanja te u bilo koje vrijeme može zatražiti preporuke od Letizie. Dok korisnik pregledava sadržaj određene stranice, Letizia započinje pretraživanje ″lokalnog susjedstva″ (eng. local neighborhood) te stranice. Prati linkove sa trenutne stranice sve dok je korisnik na toj stranici. Pri tome koristi metodu pretraživanja po širini. Čim korisnik zatvori stranicu, Letizia odbacuje trenutnu pretragu i započinje pretraživanje sljedeće stranice koju korisnik otvori. Ovaj proces se zove ″izviđanje″ (eng. reconnaissance) jer Letizia izviđa novi teritorij prije nego mu korisnik pristupi [LIEB2001].



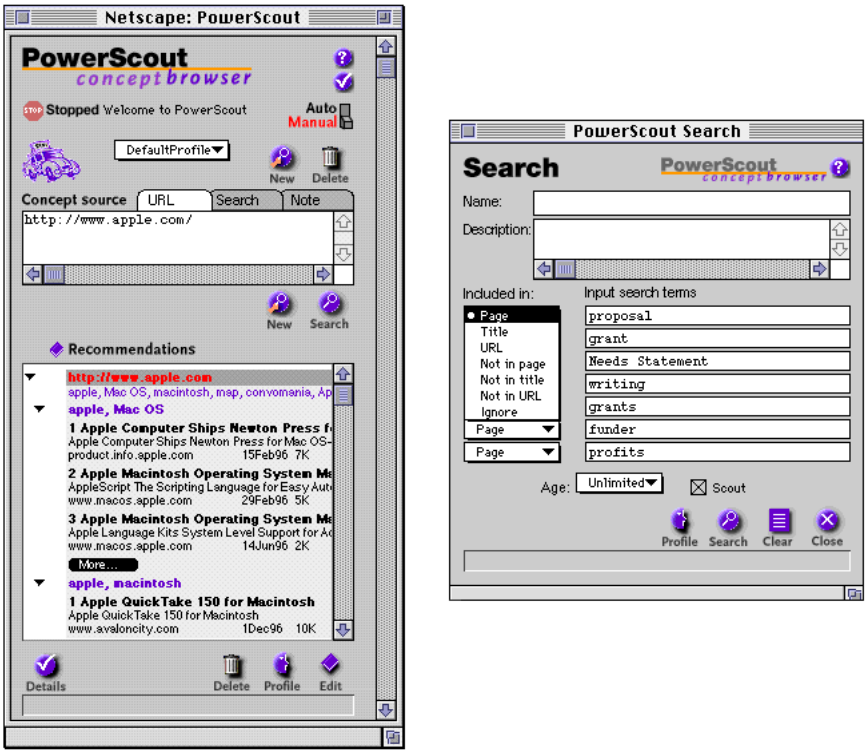
Slika20. Izgled Letizia zaslona

Na Slici20 možemo vidjeti izgled Letizia zaslona. Lijevi prozor je namjenjen za korisnikovo pretraživanje neovisno od pretraživanja agenta. Druga dva desna prozora su pod kontrolom agenta. Gornji prozor prikazuje listu stranica koje Letizia namjerava pretražiti, dok donji prozor prikazuje one stranice koje Letizia preporučuje korisniku. Korisnik bira da li će pretraživanje nastaviti koristeći preporuke Letizie ili ne. Letizia također daje i objašnjenje zašto preporučuje određeni dokument.

4.7. Powerscout

Za razliku od agenta Letizie koji pretraživanje provodi u lokalnom susjedstvu trenutne stranice, agent *Powerscout* pretražuje Web u globalu [LIEB2001]. Također prati korisnikovo ponašanje i daje pregled preporuka, ali koristi drugačiju strategiju za prikupljanje i prikaz tih preporuka, tj. pretraživač (Netscape) kao potporu tehnikama globalnog izviđanja. Powerscout koristi svoj model korisnikovih interesa kako bi sastavio i poslao upit za pretraživač dok korisnik pretražuje Web. Koristeći pretraživač za pronalazak dokumenata u ″semantičkom susjedstvu″ (eng. semantic neighborhood) trenutnog dokumenta koje korisnik pretražuje, sustav koristi drugačiju vrstu pretraživanja – pretraživanje pojma (eng. concept browsing). Ovaj termin ističe ideju pretraživanja linkova koji nisu određeni od strane autora dokumenta ali su semantički značajni za dokument koji se pregledava. Pojmovi su formulirani pomoću izvlačenja ključnih riječi iz

trenutne sranice. Svaki pojam predstavlja malo različito gledište o tome što je važno na stranici. Korisnik može brzo pregledati pojmove i odlučiti koji mu je, ako postoji, koristan. Ispod svakog pojma nalazi se kratki popis stranica sa sažecima. Klikom na pojedini naslov stranice otvara se cijela stranica u pretraživaču. Na Slici21 možemo vidjeti izgled Powerscout zaslona. Na lijevom prozoru možemo vidjeti listu preporuka grupiranih po pojmu. Desni prozor predstavlja prozor za dijalog tijekom pretraživanja (eng. search dialog box). Sastoji se od polja za uređenje teksta (eng. text edit field) za svaki od sedam termina. Kada korisnik zatraži prikaz prozora za dijalog on je već napunjen (eng. preloaded) sa sedam najznačajnijih termina sa stranice koja se upravo pretražuje. Korisnik tada može urediti termine tako da uputi pretraživač na naslove stranica, adrese, itd. Klikom na botun Search upit je generiran i poslan pretraživaču.



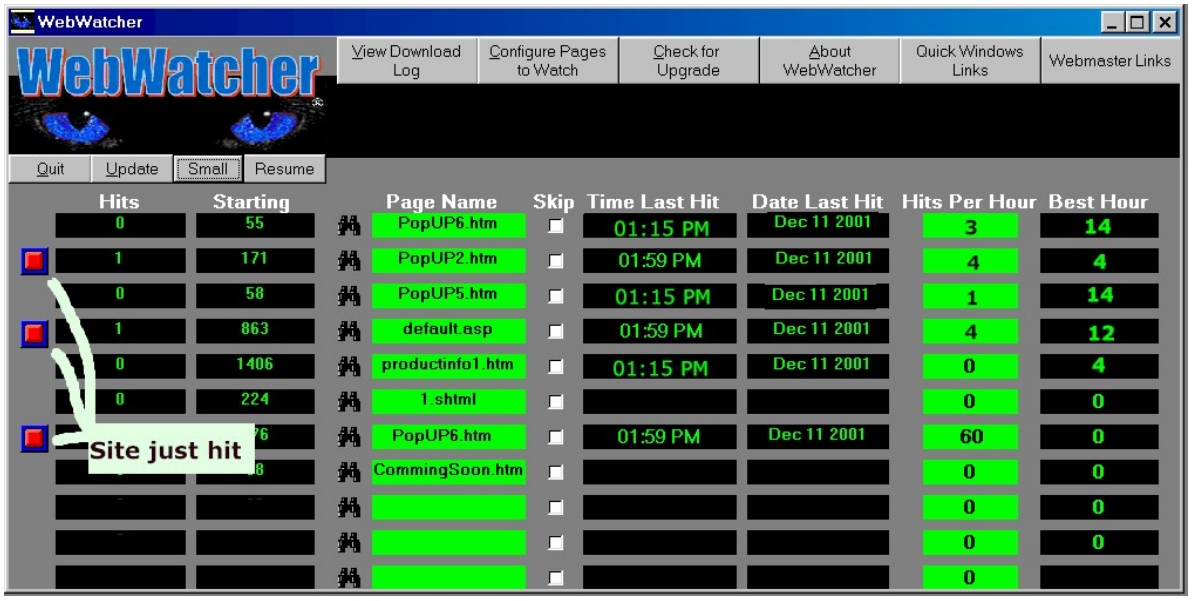
Slika21. Izgled Powerscout zaslona

Korisnikovi interesi za određeno područje predstavljeni su u profilu korisnika (korisnik može kreirati više profila kako bi opisao svoje interese). Svaki profil je sastavljen od niza poredanih termina i bilješka pretraživanja (eng. research notebook). Termini su riječi izvučene iz mnoštva različitih izvora kao što su web stranice, određena pretraživanja, tekst iz dokumenata i e-mail poruka. Bilješka pretraživanja omogućuje korisniku pristup izvorima termina, tj. originalnim URL-ovima i tekstovima.

4.8. WebWatcher

Slika22. WebWatcher ikona

WebWatcher agent također olakšava korisniku pretraživanje Web-a [JOAC1995]. Na Slici23 dan je izgled glavnog operativnog zaslona. Klikom na botun na WebWatcher serveru (Configure Pages to Watch) korisnik unosi WebWatcher agenta u pretraživanje i svoje interese opisuje ključnom riječi. Nakon toga WebWatcher vraća korisnika na stranicu od koje je započelo pretraživanje. Od tada WebWatcher prati korisnikove akcije i sugerira linkove koristeći znanje naučeno iz korisnikova ponašanja. Također nudi i niz drugih mogućnosti: naglašava linkove na trenutnoj stranici za koje smatra da su od interesa korisniku, dodaje nove linkove na temelju korisnikovih interesa, sugerira stranice povezane sa trenutnom stranicom te šalje poruke korisniku o promjenama određenih stranica. Korisnik može prekinuti WebWatcher-a u bilo koje vrijeme ovisno da li je pretraga bila uspješna ili ne.

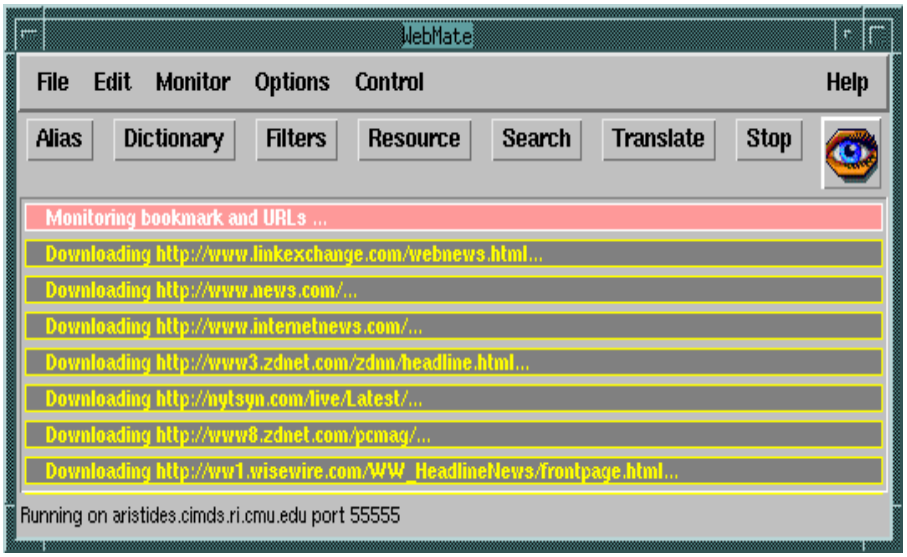


Slika23. Glavni zaslon WebWatcher-a

Više o WebWatcher agentu za pretraživanje može se saznati na stranici http://www.stevekaras.com/WW/WWmain.htm.

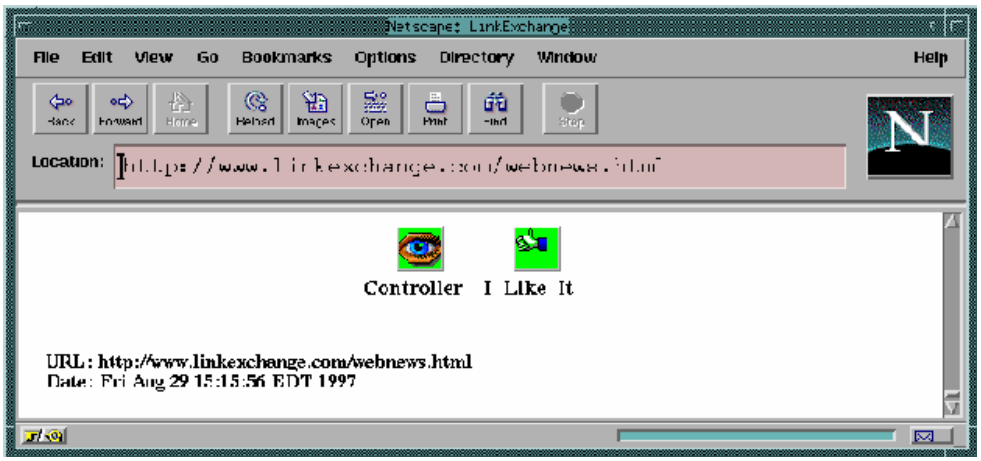
4.9. WebMate

WebMate agent omogućuje korisniku učinkovito pretraživanje Web-a [CHEN1997]. Prati korisnika tijekom pretraživanja i pruža mu informacije koje je sakupio na temelju korisnikova profila, stvorenog na temelju korisnikovog pretraživanja i odabira stranica na način ″I like it″. WebMate sučelje sastoji se od samostalnog posrednika (eng. standalone proxy) i aplet kontrolera (eng. applet contoller). Samostalni posrednik je HTTP posrednik između korisnikova Web pretraživača i Web-a (Slika24). Sve HTTP transakcije prolaze kroz WebMate koji prati korisnikovo pretraživanje i uči iz njega.



Slika24. Standalone proxy

Aplet kontoler je sučelje između korisnika i samostalnog posrednika (Slika25). Pomoću njega korisnik može izraziti svoje interese tijekom pretraživanja i ujedno koristiti pomoć od WebMate-a.



Slika25. Applet controller

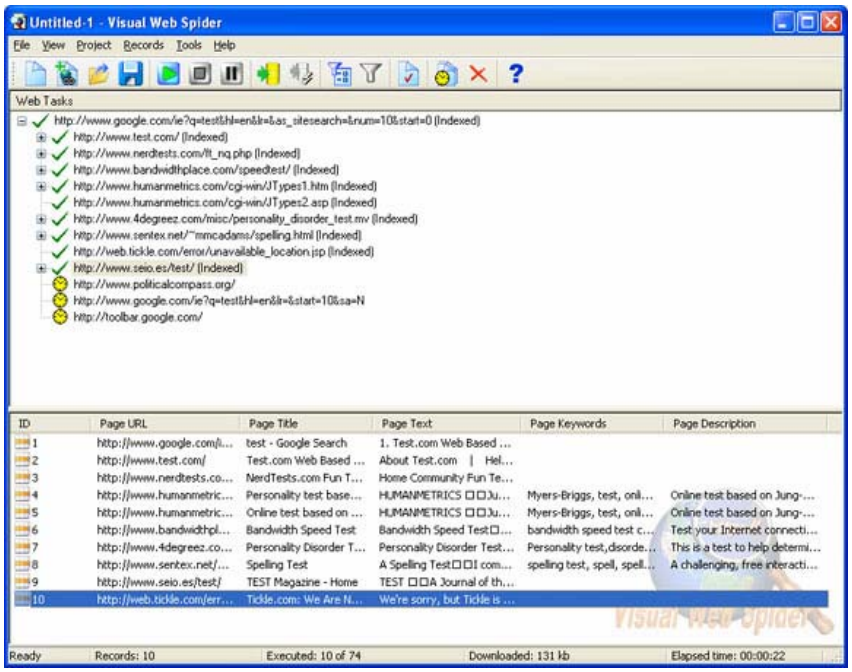
Klikom na ikonu I like it korisnik ukazuje agentu da mu odgovara trenutna stranica. Klikom na ikonu Controller otvara se izbornik sa svim naredbama WebMate-a (Slika26).

Slika26. Naredbe WebMate

Više o WebMate agentu za pretraživanje može se saznati na stranici http://www.cs.cmu.edu/~softagents/webmate.html.

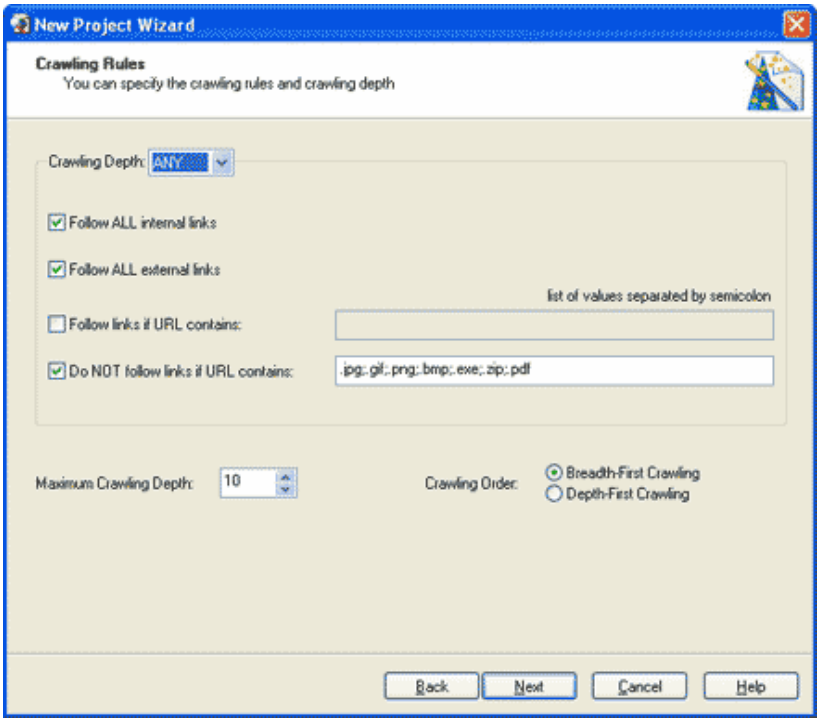
4. 10. Visual Web Spider

Visual Web Spider je agent za pretraživanje Web-a koji omogućava indeksiranje i prikupljanje specifičnih web stranica. Jednom instaliran, omogućava korisnicima pretraživanje Web-a na automatski način, indeksiranje stranica koje sadrže određenu ključnu riječ i izraze, te prikaz indeksiranih podataka u bazi podataka na vašem računalu. Glavni pozor sastoji se od dva dijela (Slika27): gornji dio sa popisom zadataka (eng. Web Tasks list) i donji dio sa popisom indeksiranih podataka (eng. indexed data).



Slika 27. Glavni prozor Visual Web Spider-a

Nakon što je program pokrenut unosi se URL adresa početne stranice, ili se prepušta programu da generira URL linkove na temelju ključne riječi ili izraza. Potom se postavljaju pravila i dubina pretraživanja ovisno o metodi pretraživanja (Slika28).



Slika 28. Kreiranje novog projekta

Nakon toga određuju s podaci za indeksiranje. Tako se mogu indeksirati sve stranice, stranice koje sadrže određenu riječ, stranice čija URL adresa sadrži određenu riječ, itd. Klikom na Start pretraživanje započinje.

5. Zaključak

Danas, na početku 21. stoljeća, zbog velike količine informacija koje nas okružuju, neophodno su nam potrebni inteligentni agenti koji omogućuju što bolje snalaženje tijekom pretraživanja Web-a. Inteligentni agenti za pretraživanje javili su se u drugoj polovici osamdesetih godina, a njihovo izučavanje je intezivirano u devedesetima jer su prepoznati kao obećavajuće rješenje za problem prikupljanja, obrade te čuvanja velike količine informacija.

Očekivanja korisnika su mnogo narasla od doba kada su se pretraživački mehanizmi tek pojavili na Internetu i kada je bilo pravo čudo kada bi nešto pronašli. Danas, ako korisnik ne dobije što traži unutar prva tri linka, često odustaje od daljnjeg pretraživanja. Zato se svakodnevno, u velikim timovima, radi na ispravljanju algoritama pretraživanja i poboljšanju samog pretraživanja. Google-ov sustav za prikazivanje rezultata se godišnje nadogradi otprilike 500 puta.

Tolika upornost, svakodnevne izmjene i dopune algoritma, veliki broj ljudi okupljenih u timu kako bi ispravljali pogreške koje se svakodnevno događaju i prikupljali pritužbe ljudi koji su razočarani rezultatima pretraživanja, ključ su za dobivanje tržišne utakmice. Naime, ljudi su postali razmaženi korisnici koji više ne traže odgovor na upit koji upišu, već zahtjevaju da im se da ono što žele. Dakle, umjesto "daj mi ono što sam upisao", sada je "daj mi ono što hoću".

Inteligentni agenti za pretraživanje upravo predstavljaju rješenje ovih problema. No njihova efikasnost ovisi o metodi pretraživanja koju primjenjuju. U radu smo naveli deset metoda pretraživanja. Svaka od njih ima prednosti i nedostatke, te ovise o samom prostoru pretraživanja. Dvije najčešće korištene metode pretraživanja su pretraživanje po dubini i pretraživanje po širini. Pretraživanje po širini uvijek osigurava pronalazak rješenja za razliku od pretraživanja po dubini. Iterativno pretraživanje po dubini je poseban slučaj pretraživanja po dubini koji ograničava dubinu pretraživanja te istodobno kombinira i metodu pretraživanja po dubini i po širini. Heurističke metode pretraživanja koriste informacije o problemu koje utječu na izbor sljedećeg čvora za pretraživanje. Tako pretraživanje najboljim prvim uvijek odabire čvor koji se čini najbližim cilju. A* pretraživanje istovremeno koristi i informacije o udaljenosti čvora od cilja i od početnog čvora te time osigurava pronalazak najboljeg rješenja čak i među više rješenja.

Također je važno napomenuti da različiti problemi pretraživanja mogu biti rješeni primjenom samo određenih algoritama pretraživanja, te da ne postoji metoda pretraživanja koja može rješiti sve probleme pretraživanja. Stoga je tijekom donošenja odluke koju metodu pretraživanja primjeniti, važno proučiti samu kompleksnost problema.

Za stvaranje inteligentnog agenta za pretraživanje Web-a postoji dosta alata i osnova na Internetu s kojima bi se trebalo upoznati. Mogućnosti za poboljšavanje inteligentnih agenata za pretraživanja su zaista brojne. Složena priroda dinamičkog Web-a znatno otežava stvaralačko

planiranje i pretraživanje, tako da bi se u inteligentne agente za pretraživanja mogle implementirati napredne pretraživačke metode i strategije planiranja.

Također, postoji jaz između načina na koji ljudi šalju zahtjev za informacijom i načina na koji računala pohranjuju informacije. Istraživanja na području obrade prirodnog jezika bi mogla znatno doprinjeti rješavanju ovog problema. Razumijevanje jezika obuhvaća razumijevanje sadržaja i konteksta, a ne samo strukturu rečenice. Ukoliko bi ljudi mogli reći agentu što žele prirodnim jezikom, a on to razumijeti, pretraživanje bi imalo puno bolje rezultate.

Pretraživanje se može maknuti sa sintaktičke osnove na konceptualnu ili semantičku osnovu dopuštajući agentu da traži ideje, a ne samo riječi. Ukoliko agent razumije koncepte (pojmove), onda se može koristiti učenjem kako bi mogao razumijeti ponašanje korisnika i usavršiti pretraživanja za pojedini korisnički tip. Istraživanja na području planiranja, pretraživanja, integracije prirodnog jezika u računala i učenja računala mogla bi znatno doprinjeti poboljšanju inteligencije agenata za pretraživanja. Inteligentni agenti za pretraživanje zasigurno predstavljaju novi izazov u informatičkom svijetu koji treba prihvatiti.

6. Literatura

[BOŽI2001] Božić, V., Intelligent software agent.
URL: http://members.tripod.com/~veliborb (25/7/08).

[CHAU2003] Chau, M., Chen, H., Personalized and Focused Web Spiders, *Web Intelligence* (ed. N. Zhong, J. Liu, Y. Yao M. Chau). Springer-Verlag, The University of Arizona, Tucson, 2003, p. 197-217.

[CHEN1997] Chen, L., Sycara, K., WebMate: A Personal Agent for Browsing and Searching, The Robotics Institute, Carnegie Mellon University, Pittsburgh, 1997.

[DALB2001] Dalbelo Bašić, B., Ribarić, S., Rješavanje problema pretraživanjem prostora stanja, *Inteligentni sustavi* (L. Budin, B. Dalbelo Bašić, N. Pavešić, S. Ribarić). MIPRO, Rijeka, 2001.

[FRAN1996] Franklin, S., Graesser, A., Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents, Institute for Intelligent Systems, University of Memphis, Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag, 1996.

[FRIS2006] Frisch, A. M., Uninformed Search, URL: http://www-course.cs.york.ac.uk/lpa/slides2006/uninformed-search.pdf (14/08/08).

[HEYL1998] Heylighen, F., Problem solving,
URL: http://pespmc1.vub.ac.be/PROBSOLV.html (15/08/08).

[JENN1998] Jennings, N. R., Wooldridge, M., Applications of Intelligent Agents, Queen Mary & Westfield College, University of London.

[JOAC1995] Joachims, T., Mitchell, T., Freitag, D., Armstrong, R., Web Watcher: Machine Learning and Hypertext, Carnegie Mellon University, Pittsburgh, 1995.

[KEND2001] Kendall, G., Intoduction to Artificial Intelligence, URL: www.cs.nott.ac.uk/~gxk/courses/g5aiai/003blindsearches/blind_searches.htm (10/09/08).

[LARI2005] Pretraživački alati,
URL: http://laris.fesb.hr/Claroline-1.3.1/INT01_SVI05/document (16/07/08).

[LIEB1995] Lieberman, H., Letizia: An Agent That Assists Web Browsing, Media Laboratory, Massachusetts Institute of Technology, Cambridge, 1995.

[LIEB2001] Lieberman, H., Fry, C., Weitzman, L., Why Surf Alone?: Exploring the Web with Reconnaissance Agents, Communications of the ACM, 2001, p. 69-75.

[PANI2000] Panian, Ž.: Bogatstvo Interneta, Strijelac, Zagreb, 2000.

[PAVK2005] Pavković, N., Marjanović, D., Bojčetić, N., Programiranje i algoritmi, skripta, Fakultet strojarstva i brodogradnje,Sveučilište u Zagrebu, Zagreb, 2005.

[PINK1994] Pinkerton, B., Finding What People Want: Experiences with the WebCrawler, URL: http://thinkpink.com/bp/WebCrawler/WWW94.html (11/09/08)

[RAJŠ2007] Rajš, I., Krajcar, S., Krpan, K., Primjena višeagentskih sustava u simulatorima tržišta električnom energijom, *Energija,* br. 6, str. 642-675, 2007.

[ROSI2000] Rosić, M., Zasnivanje sustava obrazovanja na daljinu unutar informacijske infrastrukture, magistarski rad, Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu, Zagreb, 2000.

[RUDO2004] Rudowsky, I., Intelligent Agents, Brooklyn College, Proceedings of the Americas Conference on Information Systems, New York, 2004.

[RUSS1995] Russel, S., Norvig, P., Intelligent Agents, *Artificial Intelligence: A Modern Approach*, Englewood Cliffs, Prentice Hall, 1995, p. 31-53.

[SPET1994] Spetka, S., The TkWWW Robot: Beyond Browsing, URL: http://www.cssunyit.edu/~scott/papers/TkWWWRobot/spetka.html (11/09/08).

[TEŽA2001] Težak, Đ., Pretraživanje informacija na Internetu, URL: http://prelog.chem.pmf.hr/~tezak/preinin/preinin/index.html (26/07/08).

[YANG2000] Yang, C.C., Yen, J., Chen, H., Intelligent internet searching agent based on hybrid simulated annealing, *Decision Support Systems*, Volume 28, Number 3, p. 269-277, 2000.

[YOUN1999] Youngblood, G. M., Web Hunting:Design of a Simple Intelligent Web Search Agent, URL: www.acm.org/crossroads/xrds5-4/webhunting.html (16/07/08).

[WOOL1995] Wooldridge, M., Jennings, N., Intelligent Agents: Theory and Practice, URL: www.csc.liv.ac.uk/~mjw/pubs/ker95/ker95-html.html (20/07/08).