SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Genetski algoritmi i problem N kraljica

Dražen Lučanin

Voditelj: *Domagoj Jakobović*

Zagreb, travanj, 2008.

Sadržaj

1. Uvod

U programiranju i optimiranju su velik izazov tzv. NP-problemi – problemi koji se deterministi̇čkim algoritmima mogu riješiti samo u nepolinomijalnom vremenu. Kod NP-problema brzina računala nema mnogo utjecaja na vrijeme potrebno za rješavanje problema. Iz nužnosti za učinkovitijim rješavanjem tih problema razvijena je cijela klasa heurističkih algoritama koji daju približno rješenje, ali u znatno manje vremena. U heurističke algoritme se ubrajaju i genetski algoritmi, koji se ugledaju na prirodnu selekciju u stvarnom svijetu (opstanak boljih jedinki i odumiranje lošijih) kao metodu pronalaženja rješenja.

Problem N kraljica je jedan NP-problem sa važnim primjenama na kojem će biti opisana implementacija genetskog algoritama te njegove prednosti i mane.

2. Genetski algoritmi

„Bez obzira radilo se o brojevima, simbolima ili stolićima za kavu, možemo stvoriti odgovarajući digitalni svemir koji će nam omogućiti da se te svari razmnožavaju uz nasljeđivanje i varijaciju i da postoji neki oblik odabira. Kada omogućimo događanje tih triju procesa, evolucija će se jednostavno dogoditi. A kada se u računalu dogodi evolucija, rješenja često iznenađuju."

– *Peter Bentley, „Digitalna biologija"*

Digitalni svemiri o kojima je tu riječ su u biti računalni programi, a kada oni koriste evoluciju, mehanizam preuzet iz prirode, kako bi došli do rješenja, govorimo o evolucijskim algoritmima. Evolucijski algoritmi spadaju u klasu heurističkih algoritama zbog toga što rješenje koje daju nije uvijek precizno i najbolje kao kod deterministčkih algoritama, već je približno. Svi evolucijski algoritmi dijele zajednički princip: populacija jedinki prolazi kroz neke transformacije i za vrijeme tog evolucijskog procesa jedinke se bore za opstanak.

Genetski algoritam (GA) je vrsta evolucijskog algoritma. To je tehnika pretraživanja prostora potencijalnih rješenja problema. Kako se traži najbolje rješenje, na GA možemo gledati kao na proces optimizacije. Kako ne bi morao pretražiti cijeli prostor, GA koristi umjetnu inteligenciju – genetsko nasljeđivanje i Darwinovsku borbu za opstankom.[Michalewicz, 1992.]

2.1 Povijest genetskih algoritama

Charles Darwin je autor povijesno najpoznatije teorije evolucije. Imao je povoljan teren, jer u to vrijeme biologija raspolaže velikim brojem činjenica koje potkrepljuju teoriju evolucije živoga svijeta. Tako su postignuti značajni rezultati na područjima komparativne anatomije, komparativne embriologije (K.E. Baer ustanovio sličnosti zametaka u kralježnjaka). T. Schwann 1839. otkriva jedinstvo stanične građe svih živih bića, i osniva nauku o stanici – citologiju.

Darwin je teoriju evolucije predstavio u svom glavnom djelu, „O podrijetlu vrsta posredstvom prirodne selekcije ili o održavanju povlaštenh rasa u borbi za opstanak" 1859. godine. Njegova se teorija evolucije prvenstveno oslanja na prirodni odabir, selekciju, koja podupire jedinke bolje prilagođene uvjetima okoline, dok slabije prilagođene jedinke odumiru – **opstanak najsposobnijih**.

Suvremene evolucijske teorije se najvećim dijelom oslanjaju na darvinizam, uz manje izmjene koje su rezultat nedavnih istraživanja (osnovne sile evolucije su mutacija, prirodno odabiranje i genetski posmak (engl. drift)). Tako npr. danas znamo da je molekula DNK osnovni nositelj genetske informacije. Njezinu kemijsku strukturu su 1953. godine otkrili J.D. Watson i F.H.C. Crick. Četiri dušične baze molekule DNK su nosioci jedinične informacije u prirodi, slično bitovima (0 ili 1) u digitalnim računalima. Pomoću tih se informacija „grade" živa bića i u tom obliku dijete nasljeđuje dio osobina svakog roditelja.[Božiković, 2000.]

Pojavom računala počele su se javljati ideje o izradi „umjetnog života" zasnovanog na načelima evolucije. Računalne simulacije evolucije su započele već

1954. godine radom Nilsa Aalla Barricellija, iako njegove publikacije nisu bile toliko zapažene. Ipak, tijekom 60ih godina takve su simulacije za potrebe biologije postale uobičajene. Uz to, Hans Bremermann je pisao o stvaranju populacije rješenja problema optimizacije, koja prolaze kroz rekombinaciju, mutaciju i selekciju što su elementi modernih genetskih algoritama.[Wikipedia-genetic algorithm]

Značajnija ostvarenja postižu Ingo Rechenberg i Hans-Paul Schwefel 60ih i 70ih s svojim evolucijskim strategijama kojima su mogli rješavati komplicirane inženjerske probleme i Lawrence J. Fogel są evolucijskim programiranjem koje je zapravo umjetna inteligencija zasnovana na evoluiranju konačnih automata. Konkretno genetski algoritmi postaju popularni kroz rad **Johna Hollanda** u ranim 1970ima i pogotovo njegovu knjigu „Prilagodba u prirodnim i umjetnim sustavima" iz 1975. g.[Wikipedia-genetic algorithm]

Istraživanje genetskih algoritama je ostalo uglavnom teoretsko do sredine 1980ih, kada je održana Prva međunarodna konferencija genetskih algoritama u Pittsburghu, Pennsylvaniji. S drastičnim povečanjem brzine računala, omogućena je praktična primjena nove tehnike i njezino brzo širenje.[Wikipedia-genetic algorithm]

2.2 Načela rada genetskog algoritma

2.2.1 Genetski prikaz jedinke

U genetskim algoritmima, **moguće rješenje problema je jedinka**. Na isti način kako su u prirodi jedinke u potpunosti prikazane preko slijeda dušičnih baza u molekuli DNK, tako i u genetskom algoritmu jedinke moraju biti prikazane nekim zapisom. Iz tog zapisa moramo moći jednoznačno iščitati o kojem je rješenju riječ i želimo da bude dovoljno jednostavan kako bi mogli učinkovito upravljati njime na računalu.

Mogući prikazi jedinki u GA uvelike ovise o prirodi problema. Najlogičniji izbor, koji se u početku najviše i koristio, je pokušati iskoristiti binarni broj – niz bitova. Npr. ako želimo prikazati neki realni broj na određenom intervalu s određenom preciznošću, binarni niz mora biti dovoljno dugačak da svakom od tih realnih brojeva pridruži jedisntveni slijed bitova (pridruživanje mora biti injektivno). Kako je tu očito riječ o funkciji, iz svakog slijeda možemo doći nazad do rješenja u obliku realnog broja preko inverza te funkcije (prebacivanje binarnog broja u dekadski, skaliranje na zadanu preciznost i pomak na početak zadanog intervala). Radi toga ovaj zapis jednoznačno određuje rješenje. Također, ovakvim zapisom možemo bez problema manipulirati na računalu jer je i ono samo bazirano na binarnom sustavu.

Jedinke mnogih problema ipak ne možemo prikazati na ovaj način. Problem može postavljati **ograničenja** (engl. constrains) na jedinke. U prethodnom primjeru ograničenje bi, recimo, moglo biti da brojevi moraju imati paran zbroj znamenki. Tada bi binarnim zapisom obuhvatili i puno jedinki koje ne ispunjavaju ograničenja te bi se prostor potencijalnih jedinki nepotrebno povećao. Dodatnim bi se provjerama u algoritmu moglo raditi samo s jedinkama koje ispunjavaju ograničenja, ali te bi provjere mogle biti vremenski zahtjevne. Dakle, algoritam se može prilagoditi ograničenjima, ali u oba slučaja, trajanje izvođenja algoritma će se povećati.

Alternativni zapis jedinki je često učinkovitiji – umjesto niza bitova, mogu se koristiti složenije strukture koje bolje odgovaraju danom problemu.

„Ako planina ne će doći Muhamedu, tada će Muhamed doći do planine."

[Michalewicz, 1992.]

2.2.2 Inicijalna populacija

U GA populaciju čine sve jedinke u datom trenutku. Inicijalna populacija se stvara na početku algoritma. Najjednostavnija metoda jest nasumično stvaranje određenog broja jedinki. Uglavnom je poželjno da inicijalna populacija bude vrlo raznolika jer bi se inače jedinke mogle ujednačiti i zaglaviti na nekom od lokalnih optimuma (a ne globalnih, kakve želimo pronaći).

Postoji i mogućnost stvaranja točno određene početne populacije. To se može napraviti tako da se ručno unesu početne jedinke ili da se nekim brzim algoritmom pronađe nekoliko prosječno dobrih jedinki.

2.2.3 Funkcija dobrote

Kako bi mogli odrediti koja je jedinka bolja od druge potrebna je neka mjera relativne kvalitete. Takva se mjera u GA zove funkcija dobrote. Nakon što smo stvorili populaciju, svakoj jedinki funkcijom dobrote pridružujemo neki broj – nagradu. One jedinke koje su ocijenjene kao bolje imaju veću šansu da će se neka njihova obilježja prenijeti na iduću generaciju (novu populaciju koja će nastati nakon trenutne).

Funkcija dobrote opet ovisi o detaljima samog problema. Ako npr. tražimo najkraći put između dvije točke i jedinka je neki izbor mogućih puteva, dobrota jedinke će biti obrnuto proporcionalna duljini puta.

2.2.4 Genetski operatori

Genetski operatori se primjenjuju na staru populaciju jedinki kako bi mogli ostvariti novu populaciju. Cilj nam je da se dobra svojstva stare populacije prenesu dalje, a loša svojstva uklone, ali da se pritom zadrži raznolikost jedinki. Po uzoru na prirodnu evoluciju, u GA se koriste operatori križanja (engl. crossover) i mutacije.

Križanje se odvija tako da se prvo odaberu dvije jedinke za roditelje. Odabir (ili selekcija) treba biti takav da bolje jedinke imaju više šanse od lošijih. Metoda kotača za rulet są udubljenjima za kuglicu (engl. slots) veličine proporcionalne dobroti jedinke je jedna takva metoda (kotač za rulet je zapravo nasumičan broj odabran iz nekog intervala, a svaku jedinku predstavlja jedan njegov podinterval veličine proporcionalne dobroti jedinke tako da će broj svaki put „pokazivati" na jednu jedinku).

Nakon odabira roditelja potrebno je na neki način iskombinirati njihove genetske prikaze u novi genetski prikaz djeteta koje će imati neka svojstva jednog, a neka drugog roditelja. Ako npr. jedinku prikazujemo nizom bitova, jedinka-dijete se dobiva tako da se odabere neka pozicija u nizu i svi bitovi lijevo od te pozicije se uzmu od jednog roditelja, a svi bitovi desno od drugoga. Obrnutim izborom roditelja možemo dobiti i drugo dijete istih roditelja koje će biti malo drugačije. Pri nekom drugom

prikazu jedinki možda će biti potreban drugačiji operator. Operator križanja je kvalitetan kada nova jedinka dobivena križanjem i semantički sliči roditeljima, a ne samo po svom genetskom prikazu. Tek tada dijete ima svojstva roditelja i ostvaruje se evolucija, a u suprotnom se uspješno dijete dobiva više pukim slučajem.

Mutacija je operator koji na nekom broju jedinki (određenom parametrom učestalosti mutacije) nasumice mijenja neki dio genetskog zapisa, npr. u nizu bitova komplementira nasumični bit. Time se potpuno nasumično uvodi promjena u populaciju. Ta promjena može biti i dobra i loša: ako je loša vjerojatno će kroz nekoliko generacija nestati, a ako je dobra, održat će se i tako unaprijediti cijelu populaciju te je u maloj mjeri vrlo korisna. Mutacija osigurava da se populacija ne bi skupila na mali dio ukupnog prostora mogućih rješenja problema. To si možemo vizualizirati kao da se odabire jedna jedinka i baca na drugi kraj prostora rješenja. Tako se osigurava da evolucija rješenja ne završi u slijepoj ulici (npr. ako su sve jedinke u populaciji jednake, križanje više ne uvodi nove promjene te populacija stagnira).

2.2.5 Parametri i uvjet prekida

Veličinu populacije, učestalost mutacije i postotak jedinki nove populacije koje će biti djeca jedinki iz stare populacije dobivena križanjem određuju parametri GA. . Ostale jedinke nove populacije koje se popunjavaju najboljim jedinkama iz stare populacije. O parametrima uvelike ovisi učinkovitost algoritma. Ako se npr. prečesto događa mutacija, križanje ne uspjeva doći do izražaja te konvergencija algoritma prema dobrom rješenju nestaje. Podešavanje parametara često ovisi o okolini na kojoj će se GA izvoditi, pa tako recimo veličinu populacije može ograničavati memorija računala. Optimizacija parametara GA je sama po sebi značajan problem te čak postoje i paralelni algoritmi koji ih dinamički mijenjaju. Primjer parametara može biti populacija od 100 jedinki, postotak križanja 25% i vjerojatnost mutacije 1%.

Uvjet prekida GA je određen time što je zadatak u problemu i koliko vremena se algoritam može izvoditi na računalu. Što se algoritam duže izvodi to su jedinke veće dobrote. Uvjet za prekid može biti ispunjen ako je dosegnuta tražena dobrota jedinke, ako je istekao zadan vremenski rok ili ako se dogodila izmjena zadanog broja generacija.

3. Problem N kraljica

Nakon uvoda o genetskim algoritmima, ostatak rada se bavi jednom primjenom GA na rješavanje poznatog problema N kraljica. Kako se radi o dosta starom problemu, na njemu su isprobani vrlo različiti algoritmi te služi kao dobra metoda za eksperimentalnu usporedbu učinkovitosti genetskih i drugih algoritama.

3.1 Definicija i povijest problema

Problem N kraljica jest kako postaviti N kraljica iz šaha na ploču s N×N polja tako da niti jedna kraljica ne može pojesti drugu koristeći normalne kraljičine poteze (kraljica se može micati u svim smjerovima, uključujući i dijagonale, za proizvoljan broj polja). Kraljice moraju biti postavljene tako da niti jedna ne napada drugu kraljicu, dakle, rješenje zahtjeva da ne postoje dvije kraljice koje dijele isti redak, stupac ili dijagonalu.

Problem 8 kraljica je prvi put predložio šahist Max Bezzel 1848. godine i s vremenom su mnogi matematičari, uključujući Gaussa i Georga Cantora, radili na tom problemu i njegovoj poopćenoj varijanti – problemu N kraljica. Prva rješenja je ponudio Franz Nauck 1850. godine. Nauck je proširio problem sa 8 na N kraljica. 1874. godine, S. Gunther je predložio metodu nalaženja rješenja korištenjem determinanti, a J.W.L. Glaisher je razradio ovu metodu.

Edsger Dijkstra je 1972. iskoristio ovaj problem kako bi ilustrirao moć strukturiranog programiranja. [Wkipedia-8 queens]

Problem je računski vrlo zahtjevan jer za N=8 postoji 283,274,583,552 (64x63x..x58x57/8!) načina za postaviti kraljice, ali samo 92 rješenja. Moguće je smanjiti ovaj broj korištenjem ograničenja na postavljanje kraljica tako da se zabrani postavljanje kraljice u onaj redak i stupac u kojem se već nalazi neka druga kraljica. Time se broj mogućih stanja smanjuje na 40320 (8!). Ipak, ako bi konstruirali algoritam koji pretražuje cijeli taj smanjeni prostor rješenja, njegova složenost bi bila O(n!). To znači da bi za veliki N vrijeme izvođenja bilo predugo – radi se o nepolinomijalnoj složenosti.

3.2 Važnost i primjena problema N kraljica

Osim što se radi o dobrom problemu za usporedbu učinkovitosti algoritama i o dobroj vježbi za edukacijske svrhe, za problem N kraljica postoje i druge primjene. Ranije je spomenuto da se problem najčešće rješava u varijanti sa ograničenjem svake kraljice na pojedini redak i stupac. Iako naizgled banalan, taj uvjet uvelike mijenja izradu algoritma (naročito genetskog). Da li zadržati stari algoritam i tek nakon generiranja potencijalnoga rješenja provjeravati da li je uvjet ispunjen te ga, ako treba, isključiti ili skroz promijeniti algoritam tako da pretražujemo samo rješenja koja već ispunjavaju uvjet? Očito ćemo drugim pristupom puno smanjiti prostor potencijalnih rješenja te će taj algoritam biti učinkovitiji, a i provjera uvjeta iz prve varijante bi za N kraljica (iako samo linearne složenosti) oduzimala puno vremena

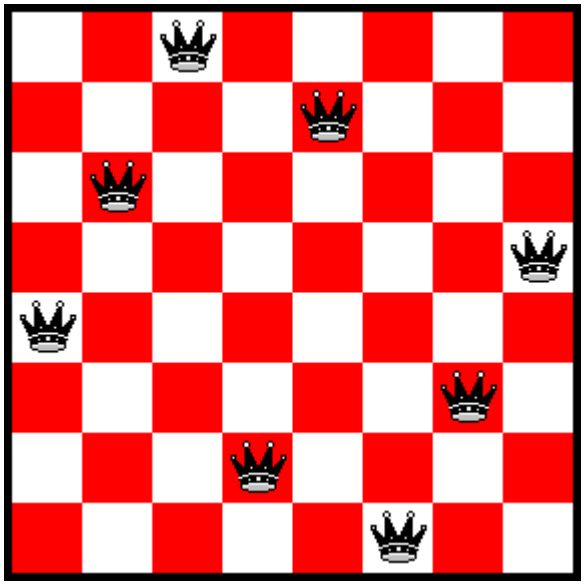
ako se treba izvoditi svako malo. Logično je stoga napraviti algoritam koji će biti prilagođen problemu sa ograničenjima, a ne obrnuto.

Puno realnih NP-problema današnjice također imaju neka slična ograničenja. Neki od tih problema su raspored velikog broja elemenata na integriranim sklopovima (engl. VLSI- very large scale integration), kontrola cestovnog i zračnog prometa, izrada rasporeda i drugi problemi uređivanja i nizanja [Homaifar, Turner, Ali, 1992.]. Kada pronađemo učinkovito rješenje za problem N kraljica, stoga, isti pristup možemo primijeniti i za ove probleme.

4. Izrada genetskog algoritma za problem N kraljica

Na isti način kao što su prethodno opisani genetski algoritmi općenito, u ovom će poglavlju biti opisan GA za rješavanje problema N kraljica.

4.1 Genetski prikaz jedinke



Slika 4.1 Konfiguracija 8 kraljica (ujedno i rješenje)

Jedinka, tj. potencijalno rješenje će u ovom slučaju biti jedna konfiguracija kraljica na ploči. To se može prikazati kao uređena n-torka gdje broj predstavlja redak, a mjesto na kojem se broj nalazi predstavlja stupac u kojem se kraljica nalazi. Konfiguraciju na slici 4.1 (gdje je N=8) bi tako bila prikazana kao (5,3,1,7,2,8,6,4).

Na te n-torke možemo gledati kao na permutacije od (1,2,3,4,5,... N) te je uvjet da svaka kraljica ima svoj redak i stupac ispunjen. Na ovaj zapis jedinke ne gledamo kao na niz bitova jer on ima drugačije semantičko značenje. Permutacija neke n-torke (gdje se isti broj se u nizu ne smije pojaviti dva puta) je apstraktniji podatak od niza bitova. Zbog toga ne će biti korišteni klasični genetski operatori nego će ih trebati prilagoditi da značenjski mijenjaju jedinke.

4.2 Inicijalna populacija

Stvaranje inicijalne populacije ne igra toliko veliku ulogu u radu GA dok god su jedinke raznolike. Najjednostavniji način jest nasumično stvoriti svaku jedinku.

4.3 Funkcija dobrote

Potreban je neki način ocjenjivanja dobrote svake jedinke. Kako je cilj da se niti jedna kraljica ne napada, onda će dobrota jedinke biti tim veća, što se manji broj

kraljica napada. Potrebno je provjeravati samo dijagonale (zbog ograničenja u prikazu jedinki koje ne dopušta napadanje u stupcu i retku). Kako ukupno ima N kraljica i svaka ima dvije dijagonale, a želimo primjerice da dobrota bude na intervalu (0,1), nagrada jedinki za nenapadanje iznosi **1/(2*N)** za svaku kraljicu koja ne napada nikoga na pozitivnoj dijagonali (dolje lijevo – gore desno) i još 1/(2*N) ako ne napada nikoga na negativnoj dijagonali (dolje desno – gore lijevo).

Uzmimo primjer jedinke (3,1,2,4) gdje je N=4. Prva kraljica na pozitivnoj dijagonali ne napada niti jednu kraljicu pa će dobiti nagradu 1/(2*4) i na negativnoj dijagonali također ne napada nikoga pa će dobiti još 1/(2*4), što ukupno daje 1/4. Druga kraljica na pozitivnoj dijagonali ne napada nikoga, ali na negativnoj dijagonali napada treću kraljicu pa će ukupno dobiti nagradu samo 1/8. Treća kraljica će dobiti 1/8, a četvrta 1/4. Kada se to sve zbroji ispada da je ukupna dobrota ove jedinke 3/4. Sada možemo uspoređivati tu jedinku sa ostalima te vidjeti koja je bolja.

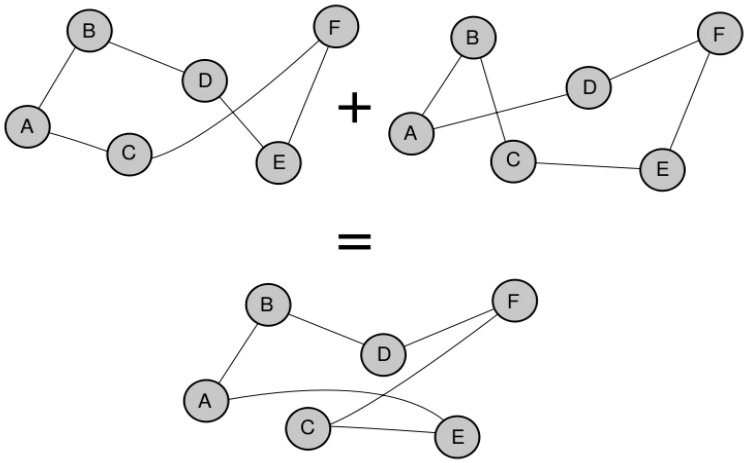
4.4 Genetski operatori

4.4.1 Operator rekombinacije bridova

Pri križanju jedinki želimo da se na neki način zadrže dobra svojstva konfiguracija roditelja. Jedan razlog dobrote jedinke može biti to što susjedne kraljice ne dijele dijagonalu. U prošlom primjeru prva kraljica je u retku 3, a njoj susjedna, druga, kraljica je u retku 1 (sto znaci da se ne napadaju) pa bi možda dijete te jedinke koje zadrži ovo susjedstvo bilo također dobro.

Ovakvo križanje se može ostvariti operatorom rekombinacije bitova [Homaifar, Turner, Ali, 1992.]. Originalno je operator zamišljen za križanje grafova, ali prikaz jedinke u problemu N kraljica se osim kao n-torka može shvatiti i kao graf s n vrhova gdje bridovi povezuju one vrhove koji su u n-torci susjedni te je učinak isti. Za oba grafa roditelja prvo je potrebno izgraditi **zajedničku** listu susjeda za svaki vrh. Za primjer na slici 4.2 su napisane liste susjeda jednadžbama 4.1.

$$
\begin{aligned}
A &= \{B, C, D\} \\
B &= \{A, C, D\} \\
C &= \{A, B, E, F\} \\
D &= \{A, B, E, F\} \\
E &= \{C, D, F\} \\
F &= \{C, D, E\}
\end{aligned}
\tag{4.1}
$$



Slika 4.2 Primjer križanja dva grafa operatorom rekombinacije bridova

Algoritam operatora rekombinacije bridova glasi:

1. *Neka je K prazna lista*
2. *Neka je N prvi vrh nasumičnog roditelja*
3. *<u>Dok</u> Duljina(K) < Duljina(Roditelj):*
   - *K := K, N (dodaj N u K)*
   - *Izbaci N iz liste susjeda svakog vrha*
   - *<u>Ako</u> (N ima nekog susjeda)*
     - *<u>onda</u> neka je N* onaj susjed od N s najmanje susjeda u svojoj listi (nasumično ako ih je više)*
     - *<u>inače</u> neka je N* nasumično odabrani vrh koji nije već u K*
   - *N := N**

Da prođemo primjer sa slike 4.2, gdje su roditelji (A,B,D,E,F,C) i (C,E,F,D,A,B). Prvo nasumično odaberemo prvog roditelja, a njegov početni vrh je A. Dodajemo ga u listu (jednadžba 4.2).

$$()\rightarrow A \tag{4.2}$$

Uklanjamo A iz listi susjeda svih vrhova i tražimo njegovog susjeda (B, C ili D) koji sada ima najkraću listu, a to je B (4.3).

$$\begin{gathered} B=\{C, D\} \\ C=\{B, E, F\} \\ D=\{B, E, F\} \\ (A)\rightarrow B \end{gathered} \tag{4.3}$$

Uklanjamo B iz svih lista i promatramo njegovu listu. Liste od C i D su iste pa biramo D (4.4).

$$\begin{gathered} C=D=\{E, F\} \\ (A, B)\rightarrow D \end{gathered} \tag{4.4}$$

Uklanjamo D. E i F su jednako dugih listi pa biramo F (4.5).

$$\begin{gathered} E=\{C, F\} \\ F=\{C, E\} \\ (A, B, D)\rightarrow F \end{gathered} \tag{4.5}$$

Uklanjamo F. C i E su jednako duge pa biramo C (4.6).

$$\begin{gathered} C=\{E\} \\ E=\{C\} \\ (A, B, D, F)\rightarrow C \end{gathered} \tag{4.6}$$

Uklanjamo C. Najkraća lista je E (4.7).

$$\begin{gathered} E=\{\} \\ (A, B, D, F, C)\rightarrow E \end{gathered} \tag{4.7}$$

Duljina djeteta je sada jednaka duljini roditelja te algoritam završava. Konfiguracija djeteta je (A,B,D,F,C,E). Bitno je primijetiti da je jedini novi brid koji smo uveli AE, a svi ostali su se nalazili kod barem jednog od roditelja. [Wikipedia-ERO]

4.4.2 Mutacija

Jedinku u problemu N kraljica mutiramo tako da zamijenimo mjesta dvjema nasumično odabranim kraljicama. U zapisu jedinke kao n-torke to izvodimo tako da nasumično odaberemo dvije pozicije te zamijenimo vrijednosti koje su na njima upisane.

4.5 Parametri i uvjet prekida

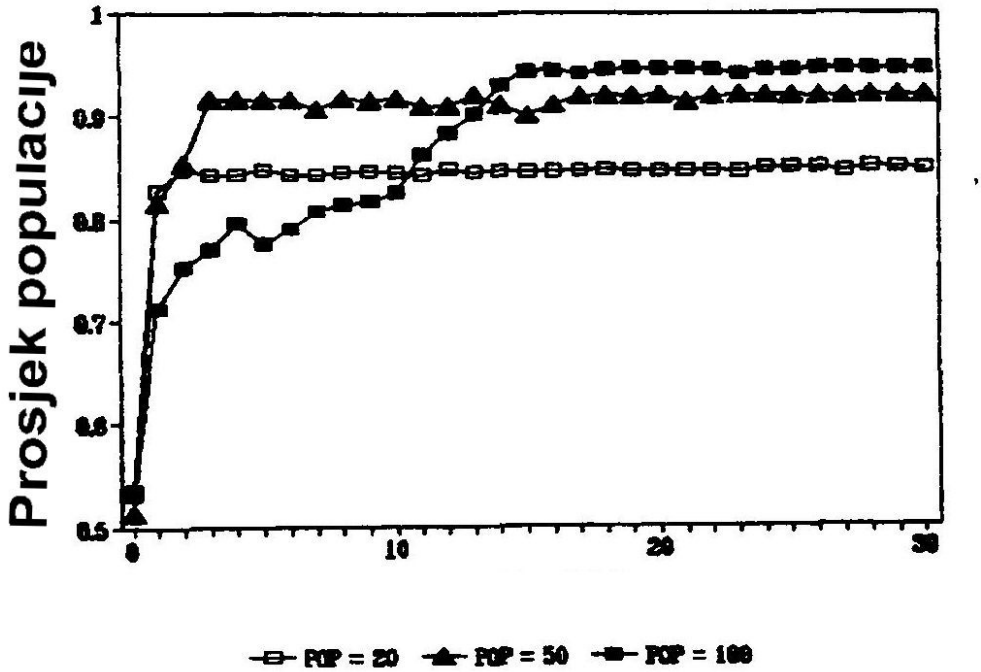
I u ovom problemu će postavljanje parametara ovisiti o okolini izvođenja programa. Veća populacija prema [Homaifar, Turner, Ali, 1992.] uglavnom daje bolje prosječne dobrote jedinki, ali ne uvijek (ovisi i o tome koji je N) što je vidljivo u tablici 4.1 i na grafu 4.5.. Oni su u svojim mjerenjima dobili najbolju prosječnu dobrotu za N=200 pri populaciji od 50 jedinki, ali za N=20 pri 20 jedinki. Iako bi možda paralelnim izvođenjem algoritma na više računala evolucija rješenja ubrzala, na jednom računalu više jedinki znači duže vrijeme izvođenja jedne generacije (računanja dobrote i stvaranja novih jedinki) te ne jamči uvijek vremenski bržu evoluciju. Zbog toga je vrlo teško odrediti optimalnu veličinu populacije.

Postotak križanja i mutacije u iznosu 25% i 1% daje dobre rezultate. Manje promjene tih parametara mogu donekle i popraviti algoritam, ali prevelike promjene ga mogu jako pogoršati (kao i kod svih GA). Iz rezultata [Homaifar, Turner, Ali, 1992.] se jasno vidi uloga mutacije na grafovima 4.3 i 4.4. Iako je mutacija tek sekundarni operator (manje važna od rekombinacije), bitna je za pravilnu konvergenciju populacije prema optimalnoj dobroti.

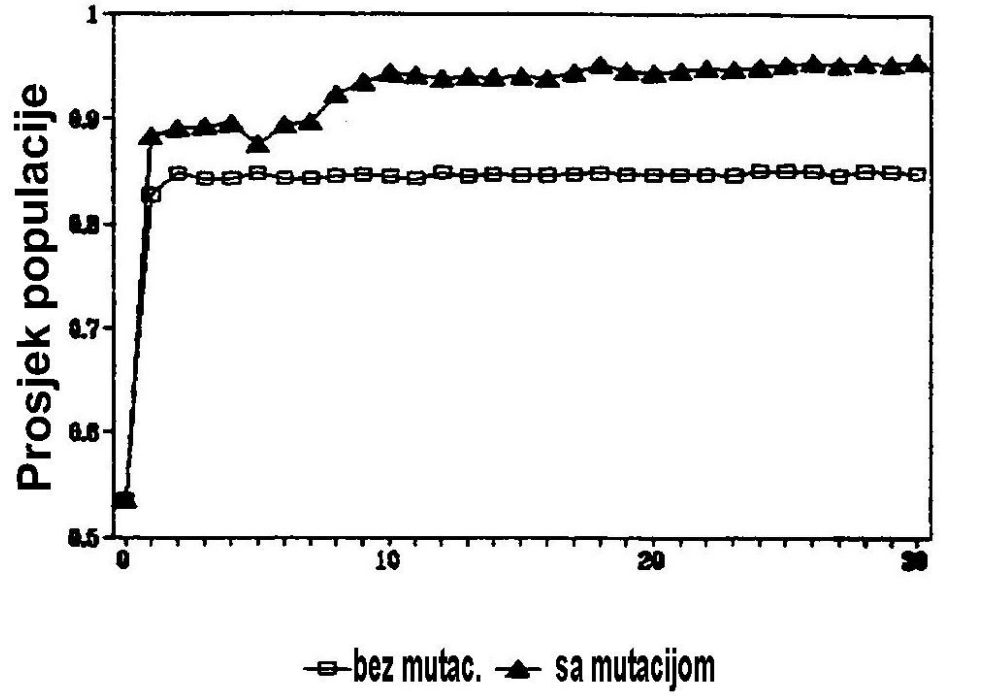
Uvjet prekida jest pronalazak jedinke dobrote 1.

Tablica 4.1: Rezultati operatora rekombinacije bridova i mutacije

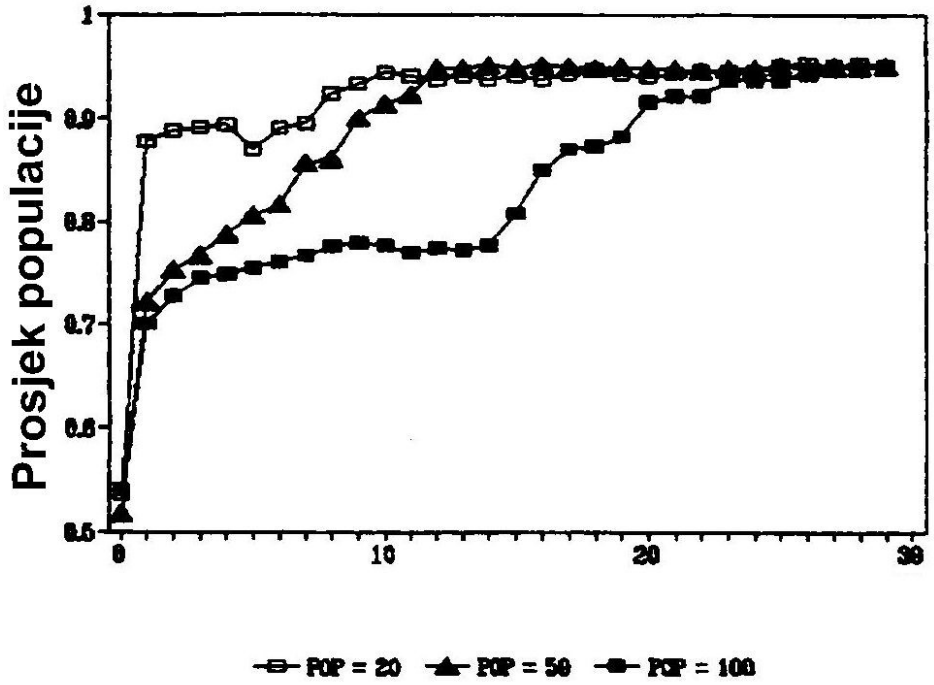
N kraljica	*Veličina populacije*	*# pokušaja do prvog rješenja*	*Prosjek dobrote populacije*
20	10	2043	.89
20	20	2141	.92
20	50	6321	.90
50	10	59,227	.957
50	20	87,975	.967
100	20	244,208	.976
200	50	340,911	.9825



Slika 4.3 Operator rekombinacije bridova bez mutacije za razne veličine populacije- prosjek dobrote na 100 uzoraka



Slika 4.4 Usporedba prosječne dobrote populacije (100 uzoraka) bez mutacije i s mutacijom



Slika 4.5 Operator rekombinacije bridova s mutacijom za razne veličine populacije- prosjek dobrote na 100 uzoraka

5. Zaključak

Svrha ovog rada bila je predstaviti genetske algoritme i njihovu korisnost pri rješavanju nepolinomijalnih problema. Na primjeru NP-problema N kraljica predstavljena je implementacija GA. Prema istraživanju [Homaifar, Turner, Ali, 1992.] a posteriori složenost tog algoritma iznosi između $O(n*\log_2 n)$ i $O(n^3)$ za serijsko izvođenje (na jednom računalu) i između $O(n*\log_2 n)$ i $O(n^2)$ za paralelno izvođenje (na više računala), što je polinomijalna složenost. U usporedbi sa deterministčkim algoritmom koji za ovaj isti problem mora pretražiti cijeli prostor mogućih rješenja i ima složenost $O(n!)$, GA, dakle, djeluje mnogo bolje. Dok deterministički algoritam u razumnom vremenu može riješiti samo probleme za N<97, GA uspješno rješava i probleme za N>200. Na današnjim računalima zapreku za velike probleme čine memorija računala i vrijeme potrebno za određivanje dobrote jedinke.

Uspješnost algoritma se može pripisati korištenju operatora rekombinacije bridova (ERO) i mutacije. ERO daje veliku sličnost djeteta i roditelja i uvodi vrlo malo novih bridova što i želimo postići. Razlog zašto je često uvođenje novih bridova loše u ovim tipovima problema je to što je jako malo novih bridova korisno, a većina njih će samo oslabiti inače dobro rješenje. ERO se prema [Wikipedia-ERO] eksperimentalno pokazuje kao bolji operator križanja od djelomično mapiranog križanja (engl. partially mapped crossover, PMX) i posrednog križanja na jednom mjestu (engl. indirect one-point crossover). Mutacija dodaje raznolikost u populaciju (ali ne prečesto) kako bi bili zastupljeni različiti dijelovi prostora potencijalnih rješenja.

Ukupna djelotvornost GA za ovakav tip problema je vrlo impresivna. GA dolazi do rješenja slijepim putovanjem po prostoru mogućih rješenja kako bi zadovoljio ograničenja postavljena zbog početne velike složenosti problema. Jedina tražena informacija je neki pokazatelj dobrote n-torki koje čine populaciju. Kako GA ne traži znanje o problemu kojega rješava, primjena na problem sa stotinama ili tisućama ograničenja je moguć. Neka područja koja od ovoga mogu imati koristi su raspored velikog broja elemenata na integriranim sklopovima, kontrola cestovnog i zračnog prometa, izrada rasporeda i drugi problemi uređivanja i nizanja.

6. Literatura

[Michalewicz, 1992.]: Michalewicz Z., Genetic Algorithms + Data Structures = Evolution Programs, 2 . izdanje. Berlin Heidelberg New York: Springer-Verlag, 1992.

[Božiković, 2000.]: , Globalni paralelni genetski algoritam. FER, 2000.

[Wikipedia-genetic algorithm]: Genetic algorithm, 10.3.2008., http://en.wikipedia.org/wiki/Genetic_algorithm, 12.3.2008.

[Wkipedia-8 queens]: Eight queens puzzle, 10.3.2008., http://en.wikipedia.org/wiki/Eight_queens_puzzle, 12.3.2008.

[Homaifar, Turner, Ali, 1992.]: Homaifar A., Turner J., Ali S., The n-queen problem and genetic algorithms. North Carolina A&T State university, 1992..

[Wikipedia-ERO]: Edge recombination operator, 14.3.2008., http://en.wikipedia.org/wiki/Edge_recombination_operator, 2.5.2008.

7. Sažetak

Upoznavanje sa genetskim algoritmima, heurističkim metodama za dobivanje zadovoljavajućeg rješenja u relativno kratkom vremenu koje se temelje na ideji evolucije preuzetoj iz prirode. Objašnjenje i kratka povijest problema N kraljica te konkretna implementacija genetskog algoritma s posebnim naglaskom na učinkovitosti operatora rekombinacije bridova. Osvrt na važnost problema i praktične primjene problema N kraljica.