Keywords: evolutionary computation, genetic algorithms, adaptive search, stochastic optimization, machine learning

Bogdan Filipič
Odsek za računalništvo in informatiko
Institut Jožef Stefan
Jamova 39, Ljubljana

Genetski algoritmi so verjetnosti preiskovalni algoritmi, zasnovani na darvinističnih načelih evolucije bioloških sistemov. Njihovo bistvo je simulirana evolucija množice dopustnih rešitev obravnavanega problema. Genetske algoritme odlikujejo robustnost, učinkovitost in nizka računska kompleksnost. Uveljavili so se v problemih preiskovanja in optimizacije ter v avtomatskem učenju. Članek predstavlja njihove osnove na primeru t.i. enostavnega genetskega algoritma, povzema njegovo teoretično analizo, obravnava implementacijo in nekatere razširitve enostavnega modela ter področja uporabe genetskih algoritmov.

GENETIC ALGORITHMS — Genetic algorithms are stochastic search algorithms based on Darwinian principles of biological evolution. They key idea of genetic algorithms is to evolve a set of candidate solutions to a given problem. Genetic algorithms turned out to be robust and efficient in finding near-optimal solutions in complex problem spaces. They have been successfully applied to search, optimization and machine learning tasks. The paper summarizes the basic principles of genetic algorithms by presenting the so called simple genetic algorithm and its theoretical analysis, discusses the implementation and some extensions of the simple model, and reviews typical applications of genetic algorithms.

1. Uvod

V računalništvu zasledimo vrsto modelov in metod, zasnovanih po zgledih iz narave. Med njimi so tako pristopi, ki temeljijo na zakonitostih iz neživega sveta, kot tudi takšni, ki posnemajo dogajanje v bioloških sistemih. Znan primer prvih je verjetnostni optimizacijski algoritem simulirano ohlajanje (angl. *simulated annealing*) (van Laarhoven in Aarts, 1987), ki je analogen modelu procesa ohlajanja snovi iz statistične mehanike. V drugo skupino sodijo npr. celični avtomati, nevronske mreže in genetski algoritmi. Skupne značilnosti metod iz te skupine so sočasno obravnavanje množice enostavnih objektov, lokalnost, nehierarhična struktura in funkcionalnost, ki ni predpisana vnaprej, temveč je rezultat interakcij med obravnavanimi objekti (Langton, 1989).

Od omenjenih metod v tem delu predstavljamo *genetske algoritme* (Holland, 1975; Goldberg, 1989a; Rawlins, 1991). Ti temeljijo na načelih naravnega izbora in zakonih genetike. Razviti so bili z namenom pojasniti adaptivne mehanizme iz narave in jih uporabiti pri načrtovanju umetnih, računalniških sistemov. Pionirsko delo na tem področju je s svojimi sodelavci na Univerzi v Michiganu opravil John Holland, čigar monografija *Adaptation in Natural and Artificial Systems* (Holland, 1975) je temelj današnjih raziskav genetskih algoritmov.

Osnovni princip genetskih algoritmov je simulirana evolucija množice rešitev obravnavanega problema. Pristop lahko uporabljamo kot preiskovalno metodo, optimizacijsko metodo ali metodo avtomatskega učenja (Goldberg, 1989a). Genetske algoritme odlikujejo robustnost, učinkovitost in nizka računska kompleksnost. Pri reševanju problemov z njimi ne potrebujemo predznanja, temveč le ustrezno kodiranje rešitev in mero njihove uspešnosti. Na osnovi lokalne informacije o kvaliteti trenutnih rešitev algoritem z adaptivnim preiskovanjem sam odkrije globalne zakonitosti problemskega prostora. Zaradi tega je evolucijski pristop posebno primeren za reševanje zahtevnih nalog, pri katerih zaradi nelinearnosti,

nezveznosti, multimodalnosti in podobnih lastnosti mnoge druge metode odpovedo.

Od tradicionalnih optimizacijskih in preiskovalnih metod, kot so analitične in numerične metode, plezalni algoritmi (angl. *hill-climbing*), naštevne metode ipd., se genetski algoritmi razlikujejo po tem, da:

- operirajo s kodami parametrov problema in ne z njihovimi vrednostmi,
- sočasno obravnavajo množico rešitev in ne ene same rešitve,
- zahtevajo le vrednost kriterijske funkcije, ne pa tudi dodatnih informacij, kot so npr. vrednosti odvodov,
- so verjetnostni in ne deterministični algoritmi.

2. Enostavni genetski algoritem

Oglejmo si najprej preprosto varianto genetskega algoritma, iz literature znano pod imenom *enostavni genetski algoritem* (Goldberg, 1989a). Ta kljub svoji enostavnosti vsebuje vse osnovne mehanizme genetskih algoritmov, tako da je z njim mogoče reševati mnoge naloge. Zaradi svoje elementarnosti je hkrati zelo primeren za teoretično obravnavo.

2.1 Osnovni koncepti

Naj bodo točke v prostoru, ki ga preiskujemo z enostavnim genetskim algoritmom, predstavljene z nizi. Tako kodirane rešitve imenujemo *osebki*, kadar želimo poudariti njihovo strukturo, pa tudi *kromosomi*. Znake v nizu imenujemo *geni*. Včasih posebej omenjamo mesto (*lokus*) gena v kromosomu in njegovo konkretno vrednost (*alel*).

Vsaki točki v prostoru preiskovanja pripada vrednost kriterijske funkcije. Genetski algoritem ne operira s kriterijsko funkcijo neposredno, pač pa jo transformira v t.i. *funkcijo uspešnosti* (angl. *fitness function*). Z njo ocenjuje osebke. Za funkcijo uspešnosti zahtevamo, da je pozitivna in definirana tako, da osebkom, ki predstavljajo boljše rešitve, priredi višjo uspešnost. (Pri maksimizacijskih problemih so torej uspešnejše rešitve z višjo, pri minimizacijskih pa z nižjo vrednostjo kriterijske funkcije.) Tem zahtevam je moč zadostiti z linearno transformacijo kriterijske funkcije v funkcijo uspešnosti. Funkcija uspešnosti je ponavadi realna.

Množica osebkov sestavlja *populacijo*. Osebke začetne populacije največkrat generiramo naključno, lahko pa zanje uporabimo tudi kako drugače dobljene rezultate, ki jih želimo z genetskim algoritmom izboljšati. Populacijo procesiramo iterativno, v korakih, imenovanih *generacije*. Iz osebkov (*staršev*) iz trenutne populacije tvorimo naslednike (*potomce*), ki pripadajo novi populaciji. Velikost populacije se pri tem ne spreminja. Za generiranje naslednikov uporabljamo različne operatorje. V enostavnem genetskem algoritmu nastopajo *reprodukcija, križanje* in *mutacija*. Ti učinkujejo na osebke v vsaki generaciji.

Postopek traja, dokler ni izpolnjen ustavitveni pogoj. Ta je lahko podan na več načinov: s številom generacij, časom izvajanja, kvaliteto rešitev, njihovo konvergenco ali s kombinacijo teh kriterijev. Najpogosteje, posebno pri testiranju, predpišemo kar število generacij. Za rezultat štejemo najboljšo rešitev, ki jo je algoritem našel med izvajanjem. Osebek, ki ustreza tej rešitvi, ne pripada nujno končni populaciji, kajti potomci niso nujno uspešnejši od svojih staršev. Enostavni genetski algoritem lahko povzamemo takole:

```
Enostavni_genetski_algoritem(g, n, pc, pm);
begin
   Inicializiraj(populacija, n);
   repeat
      Reprodukcija(populacija,nova_populacija);
      Križanje(nova_populacija, pc);
      Mutacija(nova_populacija, pm);
      populacija := nova_populacija
   until ustavitveni_pogoj(g, populacija)
end;
```

V algoritmu nastopajoči parametri imajo naslednji pomen: g je predpisano število generacij, n število osebkov v populaciji, p_c verjetnost križanja in p_m verjetnost mutacije. Vlogo zadnjih dveh bomo pojasnili skupaj z opisom operatorjev.

2.2 Operatorji

V enostavnem genetskem algoritmu nastopajo trije operatorji: reprodukcija, križanje in mutacija. Z reprodukcijo se izvaja selekcija osebkov. Preživetje

uspešnih in odmiranje neuspešnih osebkov ima za posledico konvergenco od slabših k boljšim rešitvam obravnavanega problema. S križanjem in mutacijo variiramo osebke. Za razliko od reprodukcije ta dva operatorja povzročata spremembe v genetski strukturi osebkov in ju zato imenujemo tudi genetska operatorja. Opišimo delovanje operatorjev podrobneje.

2.2.1 Reprodukcija

Reprodukcija posnema načelo naravnega izbora, t.j. preživetja uspešnejših članov populacije. Operator multiplicira osebke glede na njihovo uspešnost. Večja kot je uspešnost osebka, večja je verjetnost, da bo ta osebek prispeval določeno število potomcev v naslednjo generacijo. Ker pa se velikost populacije skozi generacije ne spreminja, manj uspešni osebki ob tem odmirajo. Reprodukcija na ta način deluje selektivno.

Sposobnost preživetja osebka je definirana z razmerjem med njegovo uspešnostjo in povprečno uspešnostjo populacije. Naj bo velikost populacije n, vsota uspešnosti vseh osebkov v njej pa $\sum f$. Potem je pričakovano število potomcev, ki jih ob reprodukciji prispeva osebek i z uspešnostjo f_i, enako $n f_i / \sum f = f_i / \bar{f}$, kjer je $\bar{f}$ povprečna sposobnost osebkov v populaciji. Z drugimi besedami: nadpovprečni osebki v boju za obstoj preživijo, podpovprečni pa propadejo.

Implementacije reprodukcije so različne. Znana je izvedba z "ruleto", na kateri posamezni odseki predstavljajo osebke, širina odsekov pa je premosorazmerna uspešnosti osebkov. Potomce določamo z zadetki na tako uteženi ruleti. Slabost tega modela je možnost precejšnjega odstopanja dobljene porazdelitve potomcev od pričakovane zaradi verjetnostnega pristopa k izbiri potomcev. To slabost odpravlja t.i. deterministično vzorčenje, pri katerem izračunamo pričakovano števila naslednikov, vsak osebek reproduciramo v številu kopij, ki je enako celemu delu pričakovanega števila potomcev, ostanek nove populacije pa zapolnimo tako, da osebke prvotne populacije razvrstimo po padajočih vrednostih decimalnega dela pričakovanega števila potomcev, nakar ustrezno število začetnih osebkov s tako dobljenega seznama prispeva po enega potomca. Pogosto se uporablja tudi kombinacija determinističnega in verjetnostnega načina, pri katerem celi del pričakovanega števila potomcev obravnavamo deterministično, decimalni del pa verjetnostno (angl. *stochastic remainder selection*).

2.2.2 Križanje

Križanje (angl. *crossover*) je genetski operator, ki skupaj z reprodukcijo največ prispeva k moči genetskih algoritmov. Operator deluje nad pari osebkov in iz njih tvori pare potomcev. Potomci v populaciji nadomeščajo svoje starše.

Križanje poteka tako, da v populaciji naključno izberemo starševska niza. Zatem, prav tako naključno, določimo *mesto križanja* (angl. *crossing site*), t.j. celo število k z intervala $[1, l-1]$, kjer je l dolžina nizov. V zadnjem koraku med nizoma izmenjamo podniza med mestoma $k+1$ in l. Opisani način križanja imenujemo *enostavno* ali *enomestno* križanje. Primer enostavnega križanja dveh binarnih nizov vidimo na sliki 1.

Starša		Potomca	
1 0 1 1 0 1	1 1 1 0	1 0 1 1 0 1	1 0 0 1
0 0 0 1 1 0	1 0 0 1	0 0 0 1 1 0	1 1 1 0

Slika 1: Enostavno križanje.

Križanje zagotavlja izmenjavo genetskega materiala med osebki, skupaj z reprodukcijo pa širjenje tistih podnizov v populaciji, ki tvorijo dobre rešitve. Običajno ne križamo vseh osebkov v populaciji, pač pa operator uporabljamo z verjetnostjo p_c, ki je parameter genetskega algoritma. Pričakovano število osebkov, ki jih v populaciji nadomestimo z njihovimi potomci, dobljenimi s križanjem, je torej np_c.

2.2.3 Mutacija

Drugi genetski operator—mutacija—je v primerjavi s križanjem drugotnega pomena. Deluje tako, da naključno spreminja vrednosti znakov v nizih. Vloga mutacije je v preprečevanju nepovratnih izgub koristnih podnizov, ki jih lahko povzročita reprodukcija in križanje. Na mutacijo samo lahko gledamo kot na naključno preiskovanje problemskega prostora.

Mutacijo ponavadi uporabljamo z zelo majhno verjetnostjo, predpisano s parametrom algo-

ritma p_m. Za razliko od križanja, kjer verjetnost pomeni pričakovani delež osebkov v populaciji, ki se bodo križali, ta parameter pri mutaciji določa pričakovani delež mutiranih genov v populaciji.

3. Matematične osnove genetskih algoritmov

Ugotovimo lahko, da je genetski algoritem zelo preprost, saj vključuje le elementarne operacije nad nizi, ki predstavljajo rešitve danega problema. V praksi se izkaže, da tako simulirana evolucija pogosto vodi k zelo dobrim rešitvam. Poglejmo zato, kako delovanje genetskih algoritmov pojasnjuje teorija. Po knjigi (Goldberg, 1989a) bomo povzeli izpeljavo osnovnega izreka o genetskih algoritmih. V ta namen najprej definirajmo pripomoček za opisovanje podobnosti med nizi, t.i. *sheme.*

3.1 Sheme

Naj bodo rešitve problema kodirane z nizi dolžine l nad abecedo A. Dodajmo ji nov znak $*$, tako da dobimo razširjeno abecedo $A^+ = A \cup \{*\}$. Znak $*$ naj ima poseben pomen: označuje naj poljuben znak iz abecede A. Sheme (angl. *schemata*) tedaj definiramo kot nize dolžine l nad abecedo A^+.

Sheme so kompakten način zapisovanja podobnosti med nizi. Nanje lahko gledamo kot na vzorce, katerim ustrezajo konkretni nizi v populaciji. Ponazorimo to s primeri. Naj bo $A = \{0, 1\}$ in $l = 4$. Shemi $0{*}{*}1$ ustreza podmnožica nizov $\{0001, 0011, 0101, 0111\}$. Shemi 1100 ustreza le niz 1100, shemi $****$ pa ustrezajo vsi binarni nizi dolžine 4.

Za nize dolžine l nad abecedo A, kjer je $|A| = k$, obstaja $(k+1)^l$ možnih shem. Po drugi strani je posamezen niz dolžine l nad abecedo A predstavnik 2^l shem, t.j. vseh nizov z $0, 1, 2, \ldots, l$ znaki $*$ namesto originalnih znakov. V populaciji n nizov dolžine l nad abecedo A je tako zastopanih med 2^l in $n2^l$ shem. Natančno število je odvisno od raznolikosti populacije.

Definirajmo še dve lastnosti shem. Imenujmo število fiksnih, t.j. od $*$ različnih znakov v shemi, *red sheme* (angl. *schema order*) in ga označimo z o. Primera: $o(01*1) = 3$ in $o(***1) = 1$. Razdaljo med prvim in zadnjim fiksnim mestom v shemi pa imenujmo *aktivna dolžina* (angl. *defining length*) sheme in jo označimo z δ. Primera: $\delta(01*1) = 3$ in $\delta(***1) = 0$.

3.2 Osnovni izrek o genetskih algoritmih

Analizirajmo sedaj vpliv reprodukcije in genetskih operatorjev križanja in mutacije na število predstavnikov določene sheme v populaciji. Opazujmo neko konkretno shemo H. Število predstavnikov sheme H v populaciji ob času t označimo z $m(H, t)$. To število je seveda različno za različne sheme in se s časom spreminja. Čas merimo v generacijah.

Vpliv reprodukcije na pričakovano število predstavnikov sheme H v naslednji generaciji je enostavno ugotoviti. Če upoštevamo, da je pričakovano število potomcev, ki jih ob reprodukciji v času t prispeva niz z uspešnostjo f_i enako $f_i / \bar{f}(t)$, kjer je $\bar{f}$ povprečna sposobnost osebkov v populaciji, lahko zapišemo

$$m(H, t+1) = m(H, t) \frac{f(H, t)}{\bar{f}(t)}, \qquad (1)$$

pri čemer je $f(H, t)$ povprečna sposobnost nizov, ki ustrezajo shemi H v času t. S tem imamo tudi za sheme zapisano selekcijsko načelo o povečevanju števila nadpovprečnih in odmiranju podpovprečnih primerkov. Ali rast števila predstavnikov uspešnih shem lahko natančneje opišemo? Predpostavimo, da velja $f(H, t) = \bar{f}(t)(1 + c)$, kjer je c konstanta. Potem enačbo (1) lahko zapišemo tudi takole:

$$m(H, t+1) = m(H, t)(1 + c). \qquad (2)$$

Začenši s časom $t = 0$ pa ob privzeti stacionarni vrednosti c dobimo

$$m(H, t+1) = m(H, 0)(1 + c)^t. \qquad (3)$$

Enačba (3) definira geometrijsko zaporedje, s katerim je opisan učinek reprodukcije na število predstavnikov sheme H. Vidimo, da reprodukcija povzroči eksponentno rast števila predstavnikov nadpovprečnih in eksponentno upadanje števila predstavnikov podpovprečnih shem v populaciji.

Vpliv križanja proučimo preko verjetnosti, da shema H "preživi" križanje, t.j. da mesto križanja ne bo med prvim in zadnjim fiksnim mestom v shemi. Mesto križanja lahko izberemo na

$l-1$ načinov, kjer je l dolžina sheme. Od tega je $\delta(H)$ križanj destruktivnih, ker "prerežejo" shemo H. Če predpostavimo enakomerno izbiranje mest križanja, je verjetnost izgube sheme zaradi križanja enaka $\delta(H)/(l-1)$. Križanje, katerega verjetnost je p_c, bo torej shema H preživela z verjetnostjo

$$p_{sc}(H) = 1 - p_c \frac{\delta(H)}{l-1}, \qquad (4)$$

ali povedano drugače, križanje najmanj destruktivno učinkuje na sheme s kratko aktivno dolžino.

Analizirajmo podobno še vpliv mutacije. Verjetnost spremembe posameznega gena zaradi mutacije je p_m, verjetnost njegovega preživetja pa $1-p_m$. Shema H bo preživela mutacijo, če nobeno od $o(H)$ fiksnih mest v njej ne bo prizadeto. Zaradi neodvisnosti mutacij je verjetnost tega dogodka p_{sm} enaka $(1-p_m)^{o(H)}$. Ker je $p_m \ll 1$, lahko poenostavimo:

$$p_{sm}(H) = 1 - o(H)p_m. \qquad (5)$$

Mutacijo torej bolj verjetno preživijo sheme nizkega reda.

Z združitvijo (1), (4) in (5) lahko sedaj opišemo skupen učinek reprodukcije, križanja in mutacije na pričakovano število predstavnikov določene sheme H v naslednji generaciji:

$$m(H,t+1) \geq m(H,t)\frac{f(H,t)}{\bar{f}(t)} \left[1 - p_c \frac{\delta(H)}{l-1} - o(H)p_m\right]. \qquad (6)$$

Tako smo dobili izrek o shemah, imenovan tudi osnovni izrek o genetskih algoritmih (Holland, 1975). Povzamemo ga takole: med evolucijo populacije z genetskim algoritmom se z generacijami eksponentno povečuje število predstavnikov nadpovprečno uspešnih shem s kratko aktivno dolžino in nizkim redom.

3.3 Hipoteza o gradnikih in implicitni paralelizem

Izrek o shemah pojasnjuje delovanje genetskih algoritmov. Namesto direktnega konstruiranja rešitev le-ti z odkrivanjem in kombiniranjem predstavnikov uspešnih shem postopoma sestavljajo čedalje boljše rešitve. Pri tem igrajo posebno vlogo nadpovprečne sheme kratke aktivne dolžine in nizkega reda. Take sheme zato imenujemo *gradniki* (angl. *building blocks*).

Procesiranje shem v genetskih algoritmih je analiziral že Holland (1975). Izpeljal je oceno števila koristnih shem, t.j. takšnih, za katere število predstavnikov v populaciji zaradi prej omenjenih lastnosti eksponentno raste. Po tej oceni je pri procesiranju n nizov $O(n^3)$ takšnih shem. Tej lastnosti genetskih algoritmov pravimo *implicitni paralelizem.*

Omenimo v zvezi s shemami še en zanimiv teoretičen rezultat. To je *funkcija shem* (angl. *schema function*), s katero Goldberg (1989b) popiše število različnih shem v populaciji. Rezultat je splošen v toliko, da ne obravnava le binarnega pač pa k-vrednostno kodiranje in možnost upoštevanja sheme šele, če je v populaciji prisotnih določeno število nizov, ki tej shemi ustrezajo. S pomočjo te funkcije avtor za serijski in paralelni genetski algoritem izpelje optimalne velikosti populacij, pri katerih se v povprečju sprocesira največje število shem na generacijo.

4. Implementacija genetskega algoritma

4.1 Kako kodirati rešitve

Spoznali smo, da v delovanju genetskega algoritma igrajo osrednjo vlogo značilni vzorci—sheme, ki se pojavljajo v nizih. Z nagrajevanjem uspešnejših nizov in njihovim kombiniranjem algoritem gradi čedalje boljše rešitve. Kako torej zagotoviti, da bo opisan proces kar najbolj učinkovit? Če se zavedamo, da algoritem procesira sheme posredno preko nizov, je odgovor na dlani: poskrbeti je potrebno za ustrezno kodiranje rešitev. To dosežemo z upoštevanjem dveh principov kodiranja: *principa pomembnih gradnikov* in *principa najmanjših možnih abeced.* Po prvem rešitve problema predstavimo tako, da v predstavitvi nastopajo za reševanje konkretnega problema smiselni gradniki, ki vodijo k dobrim rešitvam. Po drugem uporabimo za kodiranje najmanjšo abecedo, ki omogoča smiselno predstavitev problema. Uporaba skromne abecede ima za posledico bogatejšo strukturo zapisov in s tem več možnosti za uspešnost genetskega preiskovanja.

Način kodiranja izberemo v vsakem primeru

posebej. Včasih se nam sama po sebi ponuja naravna rešitev, ki obenem zadošča omenjenim principom, v nekaterih primerih pa jo je teže določiti. Kljub enostavnosti zahtev za kodiranje zanj ni predpisanega postopka, pač pa je v veliki meri stvar intuicije in izkušenj.

4.2 Kako določiti vrednosti parametrov algoritma

V genetskih algoritmih nastopa več numeričnih parametrov. V enostavnem genetskem algoritmu smo spoznali naslednje:

- število generacij,
- velikost populacije,
- verjetnost križanja in
- verjetnost mutacije.

Z različnimi razširitvami algoritma nastopijo še dodatni parametri. V zvezi s parametri se seveda pojavlja vprašanje, kako določiti njihove vrednosti, da bo algoritem kar najbolj učinkovit in bo hkrati dal dobre rezultate. Za velikost populacije npr. velja, da premajhno število osebkov ne vsebuje zadosti shem, ki bi vodile k dobrim rešitvam, v prevelikih populacijah pa je njihovo procesiranje premalo učinkovito, zaradi česar rešitve konvergirajo prepočasi. Podobno je z verjetnostjo uporabe operatorja križanja. Prenizka povzroča stagniranje preiskovanja, pri previsoki pa uspešni osebki odmirajo hitreje, kot se utegnejo reproducirati.

Spet moramo ugotoviti, da tudi za določanje vrednosti parametrov genetskih algoritmov ni pravil ali formul. Te vrednosti pri reševanju konkretnega problema najpogosteje določimo empirično, na osnovi začetnih eksperimentov. "Optimalne" vrednosti parametrov so v precejšnji meri odvisne tudi od problema samega. Na srečo pa so genetski algoritmi dokaj neobčutljivi na vrednosti parametrov. V večini primerov se namreč izkaže, da genetski algoritem približno enako deluje pri vrednostih parametrov z določenih intervalov in ne le pri točno določenih vrednostih. Za orientacijo navedimo najpogostejše vrednosti, ki jih najdemo v literaturi. Verjetnost križanja je ponavadi med 0.5 in 1. Mutacija je dosti redkejši dogodek. Njena verjetnost je ponavadi nekaj odstotkov. Običajna velikost populacije je od nekaj deset do nekaj sto osebkov. Število generacij izmed vseh omenjenih parametrov še najbolj variira, od nekaj deset do več tisoč.

Podrobnejše študije učinka parametrov algoritma so bile narejene predvsem v optimiranju funkcij (npr. De Jong, 1975; Schaffer in sod., 1989). Znanih pa je tudi nekaj načrtnih načinov določanja vrednosti parametrov. Grefenstette (1986) je v ta namen uporabil evolucijski pristop. Definiral je t.i. *meta genetski algoritem*, to je genetski algoritem, ki kot osebke obravnava genetske algoritme za reševanje danega problema, ali natančneje, nabore vrednosti njihovih parametrov. Optimira jih glede na kvaliteto in konvergenco rezultatov. Meta genetski algoritem sicer poišče dobre vrednosti parametrov, njegova slabost pa je relativna neučinkovitost. Adaptivno, vendar bolj učinkovito problem rešuje Davis (1989). Njegov genetski algoritem vsebuje različne modifikacije genetskih operatorjev, med izvajanjem pa spremlja njihov prispevek k izboljšanju rešitev in skladno z njim spreminja verjetnosti uporabe operatorjev.

5. Razširitve enostavnega genetskega algoritma

Našteli bomo nekaj razširitev enostavnega genetskega algoritma, ki omogočajo reševanje zahtevnejših problemov. Nanašale se bodo na kodiranje rešitev, funkcijo uspešnosti, genetske operatorje in paralelizacijo genetskih algoritmov.

5.1 Kodiranje rešitev

V zvezi s kodiranjem omenimo, da probleme z več parametri oz. spremenljivkami obravnavamo analogno enoparametrskim problemom. Vsakega od parametrov kodiramo z nizom, dobljene nize pa združimo v niz—kromosom, ki predstavlja rešitev problema. Genetski operatorji ponavadi obravnavajo kromosome ne oziraje se na njihovo delitev na podnize, ki ustrezajo posameznim parametrom.

Dinamično kodiranje parametrov (Schraudolph in Belew, 1992) je dodaten mehanizem, ki omogoča usmerjanje genetskega preiskovanja v obetavnejša področja problemskega prostora in izboljševanje natančnosti rezultatov. To dosežemo s prilagajanjem interpretacije rešitev med izvajanjem algoritma. Preiskovanje pričnemo na širokih intervalih in z velikim diskretizacijskim korakom, postopama pa ožimo intervale preiskovanja in zmanjšujemo korak. Predstavitev pri tem ostaja sintaktično nespremenjena.

Za kodiranje rešitev so v uporabi tudi drugačne strukture, ne le nizi. Varšek (1991) se z genetskim algoritmom uči kvalitativnih modelov, predstavljenih z drevesi, in za njihovo evolucijo uporablja posebej prirejene operatorje. Antonisse in Keller (1987) obravnavata kodiranje in operatorje za visokonivojsko predstavitev znanja, ki vključuje objekte, relacije in pravila.

5.2 Funkcija uspešnosti

Funkcijo uspešnosti pogosto *skaliramo*, tako da omejimo razmerje med uspešnostjo najboljšega in najslabšega osebka v populaciji. Namen skaliranja je preprečiti prezgodnjo konvergenco populacije, ki je rezultat prevlade močno nadpovprečnega osebka, slučajno generiranega v zgodnji fazi evolucije. S skaliranjem uspešnosti je število potomcev, generiranih med reprodukcijo, enakomerneje porazdeljeno med osebke, kar ohranja raznolikost populacije. Skalirano uspešnost f' dobimo iz uspešnosti f bodisi z linearnim skaliranjem po predpisu $f' = af + b$ ali z eksponentnim skaliranjem $f' = f^c$.

Vzdrževanju raznolikosti populacije in s tem preprečevanju prezgodnje konvergence služi tudi *porazdeljevanje uspešnosti* (angl. *fitness sharing*) (Goldberg, 1987). Bistvo postopka je v medsebojnem zmanjševanju uspešnosti osebkov glede na njihovo podobnost, kar preprečuje pretirano množenje sorodnih osebkov. Rezultat je specializacija ali uvedba vrst. Pristop je koristen pri iskanju več ekstremov multimodalnih funkcij.

S pomočjo funkcije uspešnosti lahko obravnavamo tudi omejitve, ki veljajo za rešitve danega problema. Neizpolnjevanje omejitev ovrednotimo s "kaznovanjem", t.j. ustreznim zmanjševanjem uspešnosti osebkov. Genetskemu algoritmu samemu prepustimo, da med evolucijo kot neuspešne izloči osebke, ki ne predstavljajo dopustnih rešitev. Slabost tega pristopa je v tem, da je težko razlikovati "skoraj dopustne" rešitve, pri katerih je zadoščeno večini omejitev, od tistih, ki so sicer dopustne, a imajo nizko vrednost kriterijske funkcije.

5.3 Operatorji

Posplošitev enostavnega križanja je *večmestno križanje* (angl. *multiple-point crossover*). Pri njem za par starševskih nizov naključno določimo ne le eno temveč več mest križanja, nato pa med nizoma izmenjamo vsak drugi par istoležnih podnizov. Slika 2 kaže primer trimestnega križanja.

Starša	Potomca
1 0 | 1 1 0 1 | 1 1 1 | 0	1 0 | 0 1 1 0 | 1 1 1 | 1
0 0 | 0 1 1 0 | 1 0 0 | 1	0 0 | 1 1 0 1 | 1 0 0 | 0

Slika 2: Trimestno križanje.

Pri t.i. *uniformnem križanju* (angl. *uniform crossover*) (Syswerda, 1989) izmenjava genetskega materiala med osebkoma poteka še bolj stohastično. Uniformno križanje izvedemo tako, da se za vsak par istoležnih znakov v paru starševskih nizov posebej odločimo o morebitni izmenjavi, upoštevaje pri tem predpisano verjetnost, ki je dodaten parameter algoritma.

Inverzija je genetski operator, ki se v genetskih algoritmih redkeje uporablja, združuje pa lastnosti križanja in mutacije. Z inverzijo dobimo naslednika obstoječega niza tako, da v izbranem podnizu obrnemo vrstni red znakov, kot vidimo iz primera na sliki 3.

1 0 \| 1 1 0 1 \| 1 1 1 0

1 0 \| 1 0 1 1 \| 1 1 1 0

Slika 3: Inverzija.

Elitizem je dodaten korak v genetskih algoritmih. V elitističnem genetskem algoritmu določeno število osebkov v novo generirani populaciji nadomestimo z najboljšimi iz prejšnje generacije. S tem preprečimo morebitno izgubo najboljših osebkov zaradi genetske rekombinacije.

5.4 Paralelni genetski algoritmi

Za enostavni genetski algoritem lahko ugotovimo, da je njegova časovna zahtevnost linearno odvisna od števila generacij g, velikosti populacije n in dolžine nizov l, torej $\mathcal{O}(gnl)$. Ker je med izvajanjem algoritma v splošnem potrebno oceniti gn rešitev, na čas izvajanja seveda bistveno vpliva čas potreben za oceno (evaluacijo) posamezne rešitve. Zaradi svoje zasnove pa so genetski algoritmi zelo primerni za paralelno implementacijo, s katero

lahko dosežemo dodatno pohitritev oz. povečamo število rešitev obdelanih v enoti časa.

Ideja paralelizacije je v sočasnem obravnavanju populacije na večprocesorskem računalniku, kjer vsak procesor ustreza enemu osebku ali manjši skupini osebkov. Ozko grlo za tako izvedbo predstavlja faza reprodukcije, kjer je za izračun povprečne uspešnosti populacije potrebna uskladitev vzporednih procesov. Globalnemu nadzoru se razvijalci paralelnih genetskih algoritmov običajno izognejo z uvedbo subpopulacij ali t.i. otoškega modela evolucije. Subpopulacije vsebujejo majhno število osebkov, se razvijajo neodvisno druga od druge, le občasno si izmenjajo najboljše posameznike. Implementacija je takšna, da zahteva le lokalno komunikacijo: en procesor obravnava celotno subpopulacijo, ali pa en procesor ustreza enemu osebku, subpopulacijo pa tvori skupina osebkov na sosednjih procesorjih.

Teoretično obravnavo paralelnega genetskega algoritma zasledimo v (Pettey in Leuze, 1989), v implementacijah in eksperimentalnih rezultatih pa prednjačita skupini raziskovalcev iz Bruslja (Manderick in Spiessens, 1989; Spiessens in Manderick, 1991) in St. Augustina v Nemčiji (Mühlenbein, 1989; Gorges-Schleuter, 1989).

6. Področja uporabe genetskih algoritmov

Genetski algoritmi so bili doslej že preizkušeni v optimiranju funkcij (De Jong, 1975; Bethke, 1980; Schaffer in sod., 1989) in na problemih kombinatorične optimizacije, kot sta problem trgovskega potnika in razporejanje opravil. Problema trgovskega potnika se z genetskimi algoritmi lotevajo mnogi avtorji, med njimi Goldberg in Lingle (1985), Grefenstette in sod. (1985), Oliver in sod. (1987), Fogel (1988). Za nekatere referenčne primere problema so bili prav z genetskimi algoritmi dobljeni najboljši znani rezultati, ali pa primerljivi z že znanimi, vendar ob manjši porabi računskega časa (Mühlenbein in sod., 1988; Peterson 1990).

Razporejanje opravil je v splošnem zahteven problem kombinatorične optimizacije, na katerega naletimo v proizvodnih enotah in večprocesorskih računalniških sistemih. Uporabo genetskih algoritmov pri reševanju realnih primerov tega problema najdemo v delih (Cleveland in Smith, 1989; Whitley in sod., 1989; Syswerda in Palmucci, 1991; Filipič, 1992a,b).

Drugo področje, na katerem se uveljavlja evolucijski pristop, pa je avtomatsko učenje. Tu nastopajo genetski algoritmi kot modul za generiranje pravil v t.i. *klasifikatorskih sistemih* (angl. *classifier systems*) (Booker in sod., 1989; De Jong, 1988, 1990). To so sistemi, ki se učijo sintaktično enostavnih pravil, imenovanih klasifikatorji, v stalni interakciji z okoljem. Načelo selekcije in adaptivnost, ki ju v delovanje takega sistema vnaša evolucijski pristop, omogočata sprotno prilagajanje klasifikatorjev na spremembe v okolju. Primer takega učenja je sistem za učenje opisov konceptov GABIL (De Jong in Spears, 1991). Poznani, čeprav manj pogosti, so tudi primeri učenja z genetskimi algoritmi, ki ne uporabljajo arhitekture klasifikatorskih sistemov. Med njimi omenimo učenje enostavnih računalniških programov (Cramer, 1985; Koza, 1989) in učenje kvalitativnih modelov (Varšek, 1991).

Uporabnost genetskih algoritmov je bila pokazana na nekaterih praktičnih, predvsem inženirskih optimizacijskih nalogah. V pregledih aplikacij (Goldberg, 1989a; Davis, 1991) najdemo med drugim primere vodenja dinamičnih sistemov, načrtovanja komunikacijskih omrežij in transportnih poti, razpoznavanja vzorcev ter procesiranja slik. Genetski algoritmi so se izkazali kot učinkovita optimizacijska metoda pri določanju vrednosti parametrov regulacijskega sistema (Filipič, 1992c). V delu (Davidor, 1991) je opisana genetska optimizacija trajektorij v robotiki. Končno omenimo še *genetsko programiranje*, t.j. uporabo genetskih algoritmov pri načrtovanju kompleksnih sistemov, kot so nevronske mreže, integrirana vezja in avtonomni sistemi, katerega ambiciozni cilj je umetna evolucija računalniških modulov in robotskih sistemov (de Garis, 1991).

7. Zaključek

Genetski algoritmi so interdisciplinarno področje, ki z modeliranjem evolucijskega procesa daje nov zamah raziskavam v biologiji, genetiki, antropologiji in sociologiji, hkrati pa so izsledki s tega področja uporabni v računalništvu. Uporabni so kot splošna preiskovalna metoda, ki jo odlikujejo

nizek red časovne kompleksnosti, robustnost glede vrste problemov, ki jih z njimi lahko rešujemo, in relativna neobčutljivost na vrednosti parametrov. Njihovo delovanje je presenetljivo enostavno, saj vključuje le sintaktične operacije nad kodami, ki predstavljajo rešitve obravnavanega problema. Mnogi koraki v njihovem delovanju so naključni, vendar vgrajeni mehanizmi selekcije in kombiniranja rešitev zagotavljajo konvergenco od slabših k boljšim rešitvam.

Matematična analiza delovanja genetskega algoritma se naslanja na sheme. Z njimi opišemo značilne vzorce, ki se pojavljajo v kodah rešitev, in ki jih genetski algoritem procesira posredno, ko izvaja operacije nad kodami. Po osnovnem izreku o genetskih algoritmih število predstavnikov nadpovprečnih shem v populaciji eksponentno raste, kar prispeva k generiranju čedalje boljših rešitev.

Genetski algoritmi so definirani zelo splošno, tako da načrtovanje genetskega algoritma za konkretno nalogo običajno zahteva individualni razvoj mnogih elementov algoritma. Ključni elementi, s katerimi vplivamo na učinkovitost genetskega algoritma in kvaliteto rezultatov, so kodiranje rešitev, funkcija uspešnosti, genetski operatorji in vrednosti parametrov algoritma.

Literatura

Antonisse, H. J. in Keller, K. S. (1987) Genetic operators for high-level knowledge representations. *Genetic Algorithms and Their Applications: Proc. 2nd Int. Conf. on Genetic Algorithms*, Hillsdale: Lawrence Erlbaum Associates, str. 69–76.

Bethke, A. D. (1980) Genetic algorithms as function optimizers. Doctoral dissertation, Univ. of Michigan.

Booker, L. B, Goldberg, D. E. in Holland, J. H. (1989) Classifier systems and genetic algorithms. *Artificial Intelligence 40*, 235–282.

Cleveland, G. A. in Smith, S. F. (1989) Using genetic algorithms to schedule flow shop releases. *Proc. 3rd Int. Conf. on Genetic Algorithms*, San Mateo: Morgan Kaufmann, str. 61–69.

Cramer, N. L. (1985) A representation for the adaptive generation of simple sequential programs. *Proc. Int. Conf. on Genetic Algorithms and Their Applications*, Hillsdale: Lawrence Erlbaum Associates, str. 183–187.

Davidor, Y. (1991) *Genetic Algorithms and Robotics—A Heuristic Strategy for Optimization*, London: World Scientific Publishing Co.

Davis, L. (1989) Adapting operator probabilities in genetic algorithms. *Proc. 3rd Int. Conf. on Genetic Algorithms*, San Mateo: Morgan Kaufmann, str. 61–69.

Davis, L. (ur.) (1991) *Handbook of Genetic Algorithms*, New York: Van Nostrand Reinhold.

de Garis, H. (1991) Genetic programming—GenNets, artificial nervous systems, artificial embryos. PhD Thesis, Faculty of Sciences, Free University of Brussels.

De Jong, K. A. (1975) An analysis of the behavior of a class of genetic adaptive systems. Doctoral dissertation, University of Michigan.

De Jong, K. (1988) Learning with genetic algorithms: An overview. *Machine Learning 3*, 121–138.

De Jong, K. (1990) Genetic-algorithm-based learning. V knjigi Kodratoff, Y. in Michalski, R. S. (ur.) *Machine Learning—An Arificial Intelligence Approach*, Vol. III, San Mateo: Morgan Kaufmann, str. 611–638.

De Jong, K. A. in Spears, W. M. (1991) Learning concept classification rules using genetic algorithms. *Proc. 12th Int. Joint Conf. on Artificial Intelligence*, San Mateo: Morgan Kaufmann, str. 651–656.

Filipič, B. (1992a) Enhancing genetic search to schedule a production unit. *Proc. 10th European Conf. on Artificial Intelligence ECAI-92*, John Wiley, str. 603-607.

Filipič, B. (1992b) Optimiranje proizvodnih procesov z genetskimi algoritmi. *Zbornik prve Elektrotehniške in računalniške konference ERK '92*, Portorož, zvezek B, str. 167-170.

Filipič, B. (1992c) Genetic optimization of controller parameters. *Proc. 14th Int. Conf. on Information Technology Interfaces ITI-92*, Pula, str. 173-178.

Fogel, D. B. (1988) An evolutionary approach to the traveling salesman problem. *Biological Cybernetics 60*, 139–144.

Goldberg, D. E. (1987) Genetic algorithms with sharing for multimodal function optimization. *Genetic Algorithms and Their Applications: Proc. 2nd Int. Conf. on Genetic Algorithms*, Hillsdale: Lawrence Erlbaum Associates, str. 41–49.

Goldberg, D. E. (1989a) *Genetic Algorithms in Search, Optimization and Machine Learning*, Reading: Addison-Wesley.

Goldberg, D. E. (1989b) Sizing populations for serial and parallel genetic algorithms. *Proc. 3rd Int. Conf. on Genetic Algorithms*, San Mateo: Morgan Kaufmann, str. 70–79.

Goldberg, D. E. in Lingle, R. (1985) Alleles, loci and the traveling salesman problem. *Proc. Int. Conf. on Genetic Algorithms and Their Applications*, Hillsdale: Lawrence Erlbaum Associates, str. 154–159.

Gorges-Schleuter, M. (1989) ASPARAGOS An asynchronous Parallel Genetic Optimization Strategy. *Proc. 3rd Int. Conf. on Genetic Algorithms*, San Mateo: Morgan Kaufmann, str. 422–427.

Grefenstette, J. J. (1986) Optimization of control parameters for genetic algorithms. *IEEE Trans. Syst., Man and Cybern. 16* (1), 122–128.

Grefenstette, J. J., Gopal, R., Rosmaita, B. J. in Van Gucht, D. (1985) Genetic algorithms for the traveling salesman problem. *Proc. Int. Conf. on Genetic Algorithms and Their Applications*, Hillsdale: Lawrence Erlbaum Associates, str. 160–168.

Holland, J. H. (1975) *Adaptation in Natural and Artificial Systems*, Ann Arbor: University of Michigan Press.

Koza, J. R. (1989) Hierarchical genetic algorithms operating on populations of computer programs. *Proc. 11th Joint Conf. on Artificial Intelligence*, San Mateo: Morgan Kaufmann, str. 768–774.

Langton, Ch. G. (ur.) (1989) *Artificial Life*. Reading: Addison-Wesley.

Manderick, B. in Spiessens, P. (1989) Fine-grained parallel genetic algorithms. *Proc. 3rd Int. Conf. on Genetic Algorithms*, San Mateo: Morgan Kaufmann, str. 428–433.

Mühlenbein, H. (1989) Parallel genetic algorithms, population genetics and combinatorial optimization. *Proc. 3rd Int. Conf. on Genetic Algorithms*, San Mateo: Morgan Kaufmann, str. 416–421.

Mühlenbein, H., Gorges-Schleuter, M. in Krämer, O. (1988) Evolution algorithms in combinatorial optimization. *Parallel Computing 7*, 65–85.

Oliver, I. M., Smith, D. J. in Holland, J. R. C. (1987) A study of permutation crossover operators on the traveling salesman problem. *Genetic Algorithms and Their Applications: Proc. 2nd Int. Conf. on Genetic Algorithms*, Hillsdale: Lawrence Erlbaum Associates, str. 224–230.

Peterson, C. (1990) Parallel distributed approaches to combinatorial optimization: Benchmark studies on traveling salesman problem. *Neural Computation 2*, 261–269.

Pettey, C. C. in Leuze, M. R. (1989) A theoretical investigation of a parallel genetic algorithm. *Proc. 3rd Int. Conf. on Genetic Algorithms*, San Mateo: Morgan Kaufmann, str. 398–405.

Rawlins, G. J. E. (1991) *Foundations of Genetic Algorithms*, San Mateo: Morgan Kaufmann.

Schaffer, J. D., Caruana, R. A., Eshelman, L. J. in Das, R. (1989) A study of control parameters affecting online performance of genetic algorithms for function optimization. *Proc. 3rd Int. Conf. on Genetic Algorithms*, San Mateo: Morgan Kaufmann, str. 51–60.

Schraudolph, N. N. in Belew, R. K. (1992) Dynamic parameter encoding for genetic algorithms. *Machine Learning 9*, 9–21. Tudi Technical Report LA-UR-90-275, Los Alamos National Laboratory, Los Alamos, New Mexico, 1991.

Spiessens, P. in Manderick, B. (1991) A masively parallel genetic algorithm—Implementation and first analysis. *Proc. 4th Int. Conf. on Genetic Algorithms*, San Mateo: Morgan Kaufmann, str. 279–286.

Syswerda, G. (1989) Uniform crossover in genetic algorithms. *Proc. 3rd Int. Conf. on Genetic Algorithms*, San Mateo: Morgan Kaufmann, str. 2–9.

Syswerda, G. in Palmucci, J. (1991) The application of genetic algorithms to resource scheduling. *Proc. 4th Int. Conf. on Genetic Algorithms*, San Mateo: Morgan Kaufmann, str. 502–508.

van Laarhoven, P. J. M. in Aarts, E. H. L. (1987) *Simulated Annealing: Theory and Applications*, Dordrecht: Kluwer Academic Publishers.

Varšek, A. (1991) Qualitative model evolution. *Proc. 12th Joint Conf. on Artificial Intelligence*, San Mateo: Morgan Kaufmann, str. 1311–1316.

Whitley, D., Starkweather, T. in Fuquay, D. (1989) Scheduling problems and traveling salesmen: The genetic edge recombination operator. *Proc. 3rd Int. Conf. on Genetic Algorithms*, San Mateo: Morgan Kaufmann, str. 61–69.