PD.TVZ.HR
POLYTECHNIC & DESIGN Vol. 9, No. 4, 2021.
Mrežna inačica: ISSN 2459-6302
Tiskana inačica: ISSN 1849-1995
DOI: 10.19279/TVZ.PD.2021-9-4-07

OSNOVNI PRINCIPI GENETSKIH ALGORITAMA

BASIC PRINCIPLES OF GENETIC ALGORITHMS

Vedran Mihelj[1], Aleksandar Stojanović[2]

[1]Tehničko veleučilište u Zagrebu, Vrbik 8, 10000 Zagreb, Hrvatska, Student
[2]Tehničko veleučilište u Zagrebu, Vrbik 8, 10000 Zagreb, Hrvatska

SAŽETAK

Genetski algoritam heuristična je metoda pretraživanja inspirirana biološkim procesima evolucije. Ta se metoda pokazala efikasnom za mnoge vrste problema za čije je rješavanje potrebno pretražiti veliki prostor mogućih rješenja i za koje su egzaktne tehnike pretraživanja, kao što je dinamičko programiranje, nedovoljno efikasne. U ovom radu opisani su osnovni principi rada genetskih algoritama kao što je selekcija, križanje, mutacija i funkcija dobrote zajedno s nekim područjima njihove primjene kao što su optimizacija, genetsko programiranje i gramatičko i semantičko zaključivanje. Iako je sam princip rada genetskih algoritama jednostavan, neki od problema koji ih ograničavaju u primjeni i pronalaženju optimalnog ili prihvatljivog rješenja su kompleksna i/ili neefikasna funkcija dobrote i postizanje lokalnog optimuma što ih sprečava u konvergiranju prema globalnom optimumu.

Ključne riječi: *genetski operatori, evolucijsko računarstvo, optimizacija*

ABSTRACT

Genetic algorithm is a heuristic search method inspired by biological evolution processes. That method has proven successful for many types of problems for which finding a solution requires searching a large potential solution space and for which egzact methods, such as dynamic programming, are not efficient enough. In this paper we describe basic principles of genetic algorithms such as selection, crossover, mutation and fitness function together with some application areas such as optimization, genetic programming and grammatic and semantic inference.

Although the basic principles of genetic algorithms are simple, some problems that limit their usability and finding an optimal and/or acceptable solution are complex and/or inneficient fitness function and achieving local optimum which stops them from converging towards a global optimum.

Keywords: *genetic operators, evolutionary computing, optimization*

1. UVOD
1. INTRODUCTION

Genetski algoritmi su adaptivne metode koje se mogu upotrebljavati za rješavanje problema pretraživanja i optimizacije [1]. Osnovni principi genetskih algoritama, poput selekcije, križanja, mutacije i funkcije dobrote, prvi su put precizno postavljeni u [2], a opisani su i u mnogim drugim radovima. U [3] je opisana tehnika *simuliranog žarenja* (engl. *simulated annealing*) kojom se može optimizirati globalni optimum zadane funkcije. Rad [4] obuhvaća praktičnu primjenu genetskih algoritama namijenjenu inženjerima i znanstvenicima koji ih žele primjeniti u rješavanju specifičnih problema kao što je dizajn aviona, dijagnostika višestrukih grešaka (engl. *multiple-fault diagnosis*) i sekvencijalno raspoređivanje (engl. *sequence scheduling*). U [5] je opisana metoda optimizacije parametara numeričkih optimizacijskih problema dok je u [6] opisana primjena genetskih algoritama kroz razne vrste problema. U [7] i [8] genetski algoritmi detaljno su opisani zajedno s pripadajućim teoretskim aspektima kao što je koncept *sheme* i *hipoteza građevnog bloka* (engl. *building block hypothesis*).

Genetski algoritmi zasnovani su na principu genetskih procesa bioloških organizama. Osnovna ideja je imitiranje prirodnih procesa selekcije i razmnožavanja: pojedinci koji su najuspješniji u borbi za opstanak i pronalaženju partnera imat će najviše potomaka. Karakteristike roditelja (geni) prenose se na potomke čime se mogu dobiti potomci koji su genetski „bolji“ od roditelja, što ih čini bolje prilagođenima za okolinu.

Na analogan način genetski algoritmi do rješenja dolaze takvom jednom „evolucijom“, pod uvjetom da su ispravno implementirani. U ovom kontekstu, *populacija* je skup *jedinki* (pojedinaca) koje predstavljaju jedno moguće rješenje zadanog problema. Svakoj jedinki pridružena je vrijednost koja označava to koliko je dobro rješenje koje ona predstavlja. Ta se vrijednost zove *vrijednost dobrote* (engl. *fitness score*) [9]. Jedinke s boljom vrijednošću dobrote (u odnosu na druge jedinke) imaju se prilike međusobno *križati* čime se dobivaju *potomci* koji sadrže neke karakteristike oba roditelja. Isto tako, i jedinke s lošijom vrijednošću dobrote mogu se međusobno križati, zavisno od vrste selekcije upotrebljene za odabir jedinki za križanje (kao što je *turnirska selekcija* [1]). One jedinke koje ne uđu u proces križanja eventualno „izumru“.

Na ovaj se način dobivaju nove *generacije* koje najčešće sadrže više dobrih gena pa se time, kroz više generacija, dobri geni sve više šire. Time se efektivno pretražuje prostor mogućih rješenja koja imaju najviše izgleda da budu jedno od optimalnih rješenja.

Dobro napravljen genetski algoritam tako će se kroz više generacija „približavati“ odnosno *konvergirati* prema optimalnom rješenju. Genetski algoritmi ne garantiraju uvijek pronalaženje rješenja. Primjerice, rješenje nekada nije moguće pronaći u prostoru svih mogućih rješenja (globalnom prostoru) zbog pojave lokalnih optimuma (algoritam konvergira u samo jednom ograničenom području u kojemu ne postoji rješenje).

2. OSNOVNI PRINCIPI
2. BASIC PRINCIPLES

Rad genetskog algoritma temelji se na četiri osnovne operacije [10]:

- Inicijalizacija
- Selekcija
- Križanje
- Mutacija

Zadnje tri operacije dio su „reprodukcije“ jedinki. Reprodukcija je jedan od koraka genetskog algoritma u kojem se kromosomi dvaju „roditelja“ kombiniraju i/ili modificiraju da bi proizveli potomke za iduću generaciju.

Selekcija je postupak odabira roditelja. Ovisno o odabranom selekcijskom mehanizmu (kao što je turnirska selekcija), roditelji se mogu birati slučajnim odabirom koji favorizira one s boljom vrijednošću dobrote. Na taj način „dobre“ jedinke biti će izabrane češće od „lošijih“ [1] [10].

1. Počni s nasumično generiranom populacijom od n kromosoma duljine l (to su kandidati rješenja problema)
2. Izračunaj funkciju dobrote $f(x)$ za svaki kromosom x u populaciji
3. Ponavljaj sljedeće korake sve dotle dok nije generirano n potomaka:
   a) Postupkom selekcije (npr. turnirska selekcija) generiraj skup pojedinaca za križanje.
   b) S vjerojatnošću p_c (vjerojatnost “križanja”) križaj odabrani par kromosoma na nasumično odabranoj točki (s uniformnom vjerojatnošću) da bi se dobila dva potomka.
   c) Mutiraj pojedine gene potomaka s vjerojatnošću p_m (vjerojatnost mutacije) i smjesti rezultirajući kromosom u novu populaciju.
   Ako je n neparan jedan novi član populacije može biti odbačen slučajnim odabirom (ovo vrijedi za proces križanja gdje uvijek treba biti paran broj roditelja).
4. Prenesi određeni postotak najboljih jedinki iz prethodne generacije u sljedeću.
5. Zamijeni trenutnu populaciju s novom populacijom.
6. Idi na korak 2.

Slika 1 *Princip rada genetskog algoritma. [10]*

Figure 1 *The principle of operation of genetic algorithms. [10]*

Dva glavna mehanizma reprodukcije su *križanje* i *mutacija*:

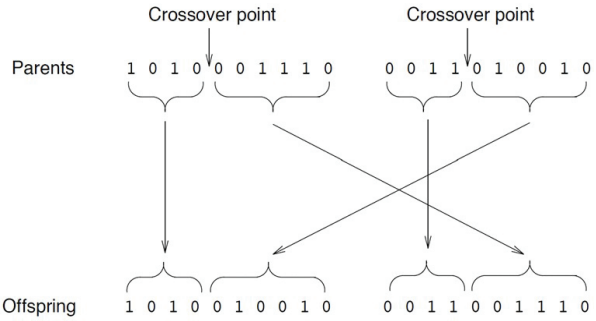
Križanje, u osnovnom obliku kako je prikazano na slici Slika 2, uzme dva roditelja (odabrana selekcijom) i podijeli njihove kromosome na dva dijela. Drugi dio oba kromosoma tada se zamijeni s onim od drugog roditelja čime se dobiju dva nova potomka (engl. *offspring*). Križanje u pravilu ovisi o problemu, a najčešće se izvodi na način da novi potomci imaju vjerojatnost 50% za svaki gen da ga naslijede od jednog ili drugog roditelja. Vjerojatnost nasljeđivanja gena također može biti ovisna o kvaliteti pojedinog roditelja gdje roditelji s boljom vrijednosti funkcije dobrote imaju proporcionalno veću vrijednost da se upravo njihov gen naslijedi.

Mutacija se izvodi nad pojedinim genima potomaka. Ona s malom vjerojatnošću (kao što je 0.001) modificira jedan ili više gena slučajnim odabirom u svrhu dodavanja raznolikosti u populaciji i eventualno ranom sprečavanju lokalnog optimuma. Slika 3 prikazuje mutaciju petog gena kromosoma.

Slika 1 prikazuje općeniti princip rada genetskog algoritma. Svaka iteracija ovog postupka daje jednu novu populaciju. Nakon generiranja potrebnog broja populacija dobije se populacija (posljednja generacija) koja sadrži jednu ili više jedinki s visokom vrijednošću dobrote. Svaka takva jedinka predstavlja jedno moguće rješenje zadanog problema.

2.1. KODIRANJE
2.1. ENCODING

Problem i rješenje genetskog algoritma prikazuje se skupom gena. Ti se geni povezuju u jedan niz vrijednosti koji se zove *kromosom* ili *jedinka*. U radu [2] autor je pokazao da je za neke probleme gene najbolje prikazati kao binarni niz, ali postoje i druge mogućnosti [11]. Na primjer, ako je problem koji rješavamo maksimiziranje neke funkcije *F(x, y, z)* onda za svaki parametar ove funkcije možemo koristiti 10 binarnih znamenki čime bi dobili kromosom koji se sastoji od 30 binarnih znamenki [1].



***Slika 2** Primjer križanja. [1]*

***Figure 2** Crossover example. [1]*

U kontekstu genetskih algoritama skup gena koji predstavljaju neki kromosom naziva se *genotipom*, dok se konkretan rezultat naziva *fenotipom*. Vrijednost dobrote neke jedinke ovisi o karakteristikama fenotipa.

2.2 VRIJEDNOST DOBROTE
2.2 FITNESS VALUE

Da bi genetski algoritam odredio koliko je neka jedinka odnosno kromosom „dobar“, to jest koliko je dobro rješenje koje on predstavlja, upotrebljava se funkcija dobrote. Ta funkcija daje broj koji predstavlja vrijednost dobrote zadanog kromosoma. Ova funkcija mora biti definirana specifično za problem koji se rješava genetskim algoritmom. Primjerice, ona može biti prilično jednostavna ako tražimo minimum ali maksimum neke funkcije s nekoliko parametara, ali će biti složenija za probleme kao što je kombinatorna optimizacija gdje broj parametara može biti puno veći. Primjerice, neka je problem maksimiziranje funkcije

$$f(x) = x+|sin(32x)|$$

za vrijednosti parametra x između 0 i π. Ovdje su potencijalna rješenja vrijednosti za parametar x koja se mogu prikazati kao niz binarnih znamenki (0 i 1) koje predstavljaju realne brojeve. Funkcija dobrote tada bi taj niz pretvorila u realni broj i evaluirala funkciju za taj x. Dobrota tog niza binarnih brojeva tada bi bila vrijednosti funkcije f za tu vrijednost parametra x [12].

Primjer

U ovom dijelu demonstriran je princip rada genetskih algoritama na jednom jednostavnom primjeru [10]. Neka je duljina *l* binarnog niza 8, funkcija dobrote *f(x)* neka daje broj bitova s vrijednošću 1 u binarnom nizu, veličina populacije *n* neka je 4,

Slika 3 *Primjer mutacije. [1]*

Figure 3 *Mutation example. [1]*

Tablica 1. *Inicijalna populacija.*

Table 1. *The initial population.*

Oznaka kromosoma	Kromosom	Dobrota
A	00000110	2
B	11101110	6
C	00100000	1
D	00110100	3

Tablica 2. *Nova generacija nakon selekcije, križanja i mutacije.*

Table 2. *New generation after selection, crossover and mutation.*

Oznaka kromosoma	Kromosom	Dobrota
E'	10110000	3
F	01101110	5
C	00100000	1
B'	01101110	5

neka je $p_c = 0.7$, a $p_m = 0.001$. Ove vrijednosti za *l* i *n* su male radi jednostavnosti demonstracije.

Na osnovu funkcije dobrote može se zaključiti da je u ovom primjeru cilj dobiti binarni niz sa što većim brojem jedinica.

Sada prva populacija (nasumično generirana) može izgledati kao u tablici Tablica 1.

Da bi mogli primjeniti operacije križanja i mutacije prvo se mora napraviti selekcija jedinki. Jedna vrsta selekcije je selekcija proporcionalna dobroti kod koje je broj puta koji je jedna jedinka odabrana za reprodukciju jednak njenoj dobroti podijeljenoj s prosječnom dobrotom cijele populacije. Jedna jednostavna metoda kojom se ovakva selekcija može implementirati je *rulet-uzorkovanje* [7]. Konceptualno, ideja te metode je da se za svakog pojedinca odredi površina kotača ruleta koja odgovara njegovoj dobroti. Kada se rulet zavrti i kuglica stane na jednoj od površina odabere se jedinka kojoj je pridružena ta površina. Ako je $n = 4$ kotač bi se zavrtio četiri puta čime bi se dobile jedinke za križanje. Na taj način što se više puta zavrti kotač to je odabir pojedinačnih jedinki bliže onom očekivanom.

Na ovaj način parovi se biraju s vjerojatnošću p_c i križaju čime se dobijaju dva potomka ili se prenose u novu generaciju nepromijenjeni. Pretpostavimo da su u ovom primjeru za križanje odabrani B i D i da su njihovim križanjem iza prvog bita dobiveni potomci E = 10110100 i F = 01101110. Isto tako, u idućoj selekciji jedinke B i C se ne križaju nego su njihovi potomci kopije istih tih roditelja. Nadalje, svaki je potomak izložen mogućnosti mutacije s vjerojatnošću p_m. Neka je, stoga, jedinka E mutirana na šestom bitu tvoreći tako E' = 10110000, potomci F i C ostaju nemutirani, a potomak B je mutiran na prvom bitu tvoreći B' = 01101110. Sada je dobivena nova generacija prikazana u tablici Tablica 2. Iako je u ovoj populaciji nestala jedinka s dobrotom 6, prosječna dobrota porasla je s 12/4 na 14/4. Kroz veći broj generacija eventualno bi dobili niz koji se sastoji samo od jedinica.

3. GENETSKI ALGORITMI U RJEŠAVANJU PROBLEMA

3. GENETIC ALGORITHMS IN PROBLEM SOLVING

U ovom dijelu opisano je nekoliko vrsta problema za koje se primjenjuju genetski algoritmi.

3.1. KOMBINATORNA OPTIMIZACIJA
3.1 COMBINATORIAL OPTIMIZATION

Kombinatorna optimizacija bavi se pronalaženjem odgovarajućeg poretka diskretnih objekata. [1] Neki od tipičnih problema koji su predmet proučavanja kombinatorne optimizacije su *problem putujućeg trgovca* [13], [14], [15]. Ovdje je cilj u neusmjerenom grafu pronaći optimalni put koji prolazi kroz sve čvorove samo jedanput koji završava u polaznom čvoru [16].

Još jedan optimizacijski problem je kako optimalno smjestiti određeni broj predmeta u neki ograničeni prostor (engl. *bin packing*). Neki radovi u kojima se ovaj problem proučava sa stajališta genetskih algoritama su [17] i [18]. Jedna primjena ovog problema je raspored VLSI integriranih krugova [19].

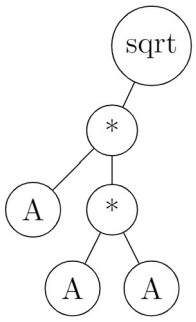
Cilj problema *raspoređivanja poslova* je kako efikasno raspodijeliti skup resursa (strojeva, prostorija, ljudi, opreme) da bi odradio neki skup poslova kao što je, primjerice, izrada dijelova za neki stroj. Ovdje je problem u tome što postoje ograničenja; na primjer, jedna te ista prostorija ne može biti rezervirana za dvije različite svrhe u isto vrijeme. Ovdje se može optimizirati najbrža izrada poslova ili najmanje neiskorištenog vremena ili prostora i slično [20], [21], [22], [23].

3.2. GENETSKO PROGRAMIRANJE (GP)
3.2 GENETIC PROGRAMMING (GP)

Cilj genetskog programiranja je dobiti računalni program „evolucijom“, na sličan način na koji bi dobili niz koji se sastoji samo od jedinica u primjeru iz prethodnog dijela. Neki od značajnijih radova u ovom području su [24] i [25]. U tim se radovima izvorni kôd prikazuje kao stablo kroz upotrebu programskog jezika Lisp. Na primjer, izraz $\sqrt[3]{a^3}$ u Lispu se piše *(SQRT (* A (* A A)))*. To se može prikazati *sintaksnim stablom* kao na slici Slika 4. U ova dva rada kandidati rješenja prikazani su na ovaj način umjesto niza bitova.

Čvorovi kao što su *sqrt* i * nazivaju se *funkcijama*, a čvorovi označeni s *A terminalima*. Genetski algoritam sada radi na ovaj način [10]:

1. Odaberi skup mogućih terminala i funkcija za program.
2. Generiraj početnu populaciju slučajnih stabala (koji moraju biti sintaksno ispravni) koristeći skup mogućih funkcija i terminala.
3. Izračunaj vrijednost dobrote za svaki program njegovim pokretanjem nad skupom testnih vrijednosti.



Slika 4 *Sintaksno stablo izraza (sqrt (* A (* A A))). [10]*
Figure 4 *Syntax tree for the expression (sqrt (* A (* A A))). [10]*

4. Primjeni operacije selekcije, križanja i mutacije. Križanje se sastoji od toga da se za svakog roditelja nasumično odaberu podstabla koja se onda zamijene.

Koraci 3 i 4 ponavljaju se za određeni broj generacija.

3.3. GRAMATIČKO I SEMANTIČKO ZAKLJUČIVANJE
3.3. GRAMMAR AND SEMANTIC INFERENCE

Cilj gramatičkog zaključivanja je dobiti gramatičku strukturu na osnovu skupa uzoraka [26], [27]. Ispod je prikazano nekoliko primjera jezika DESK [28]:

1. print a
2. print 23
3. print a + 23
4. print a + b + c
5. print a where a = 23

6. print 23 where b = 11
7. print 23 + c where c = 28
8. print 23 + 11 where c = 28
9. print a where a = 23; a = 28
10. print 28 where a = 23; b = 11
11. print 1 + 2 where b = 23; a = 5
12. print a + b + c where a = 1; b = 2; c = 3

Postupkom gramatičkog zaključivanja dobivena je sljedeća beskontekstna gramatika:

N1 → print N3 N5
N2 → + N3 | ϵ
N3 → num N2 | id N2
N4 → ; id = num N4 | ϵ
N5 → where id = num N4 | ϵ

U ovom primjeru gramatika je ispravno definirana u smislu da obuhvaća sve gore navedene uzorke. Međutim, neki od tih uzoraka nisu semantički ispravni. Na primjer, uzorci 1, 3 i 4 koriste nedefinirane varijable *a*, *b* i *c*, dok je u uzorku 9 varijabla *a* definirana dva puta. Problem je u tome što gramatičko zaključivanje obuhvaća samo sintaksu, ali ne i semantiku. U [28] i [29] opisano je semantičko zaključivanje upotrebom *atributnih gramatika* [30] koje pored definicije sintakse uključuju i semantička pravila. Kod takvog zaključivanja cilj je automatski generirati prevodilac ili interpreter na osnovu zadanih uzoraka.

Pored gore opisanog ispod su ukratko navedena još neka područja primjene genetskih algoritama:

- Ekonomija – analiza raznih ekonomskih modela kao što je teorija igara i procjenjivanje vrijednosti imovine;
- Neuronske mreže – treniranje rekurentnih i drugih vrsta neuronskih mreža.
- Problemi rasporeda (engl. *scheduling*) – pronalaženje prihvatljivog rasporeda s raznim ograničenjima (kao što je raspored sati);
- Strojno učenje – primjena *genetski zasnovanog strojnog učenja* (engl. *genetics-based machine learning* ili *GBML*);
- Analiza DNA – određivanje strukture DNA upotrebom spektrometrijskih podataka.

Ovo nije iscrpan popis primjene genetskih algoritama, ali pokazuje njihovu široku primjenu i u problemima koji nisu tradicionalno optimizacijske prirode.

4. ZAKLJUČAK
4. CONCLUSION

Iako je osnovni princip rada genetskih algoritama relativno jednostavan, detalji rada takvih algoritama zavise od problema koji se njima rješava. Način selekcije, križanja, mutacije i specifikacije funkcije dobrote te strategija izbjegavanja lokalnog optimuma ovise o specifičnostima problema koji se rješava i veličini prostora pretraživanja. Nadalje, način prikaza kromosoma (kodiranje) važan je element u dizajnu genetskog algoritma. Još jedan važan faktor u primjeni genetskih algoritama je i način implementacije; primjerice, paralelnom implementacijom i/ili adekvatnom upotrebom odgovarajućih struktura podataka može se ubrzati generiranje novih populacija. U daljnjem radu fokusirat ćemo se na problem rješavanja performansi pri pretraživanju velikih prostora za pretraživanje implementacijom raznih strategija lokalnih pretraživanja, odnosno korištenjem memetskih algoritama [31].

5. REFERENCE
5. REFERENCES

[1.] D. Beasley, D. R. Bull i R. R. Martin, »An Overview of Genetic Algorithms: Part 1, Fundamentals,« University Computing, svez. 15, br. 2, pp. 58-69, 1993.

[2.] J. H. Holland, Adaptation in Natural and Artificial Systems, MIT Press, 1975.

[3.] L. Davis, Genetic Algorithms and Simulated Annealing, Pitman, 1987.

[4.] L. Davis, Handbook of Genetic Algorithms, Van Nostrand Reinhold, 1991.

[5.] J. J. Grefenstette, »Optimization of control parameters for genetic algorithms,« IEEE Trans SMC, svez. 16, pp. 122-128, 1986.

[6.] J. J. Grefenstette, »Genetic algorithms and their applications,« u A. Kent and J. G. Williams, editors, Encyclopaedia of Computer Science and Technology, Marcel Dekker, 1990, pp. 139-152.

[7.] D. E. Goldberg, Genetic Algorithms in search, optimization and machine learning, Addison Wesley, 1989.

[8.] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer Verlag, 1992.

[9.] M. Davarynejad, M.-R. Akbarzadeh-T i N. Pariz, »A novel general framework for evolutionary optimization: Adaptive fuzzy fitness granulation,« u IEEE Congress on Evolutionary Computation, 2007.

[10.] M. Mitchell, An Introduction to Genetic Algorithms, MIT Press, 1996.

[11.] D. Beasley, D. R. Bull i R. R. Martin, »An overview of genetic algorithms: Part 2, research topics,« University Computing, svez. 15, br. 4, pp. 170-181, 1993.

[12.] M. Mitchell, »Genetic Algorithms: An Overview,« Complexity, svez. 1, br. 1, pp. 31-39, 1995.

[13.] D. E. Goldberg, »Alleles, loci, and the TSP,« u J. J. Grefenstette, editor, Proceedings of the First International Conference on Genetic Algorithms, pages 154-159, Lawrence Erlbaum Associates, 1985.

[14.] M. Gorges-Schleuter, »ASPARAGOS: an asychronous parallel genetic optimization strategy,« u J. D. Schaffer, editor, Proceedings of the Third International Conference on Genetic Algorithms, pages 422-427, Morgan Kaufmann, 1989.

[15.] G. E. Liepins, M. R. Hilliard, M. Palmer i M. Morrow, »Greedy genetics,« u J. J. Grefenstette, editor, Proceedings of the Second International Conference on Genetic Algorithms, pages 90-99, Lawrence Erlbaum Associates, 1987.

[16.] T. H. Cormen, C. E. Leiserson, R. L. Rivest i C. Stein, Introduction to Algorithms, The MIT Press, 2009.

[17.] L. Davis, »Applying adaptive algorithms to epistatic domains,« u 9th Int. Joint Conf. on AI, pages 162-164, 1985.

[18.] K. Juliff, »Using a multi chromosome genetic algorithm to pack a truck,« Technical Report RMIT CS TR 92-2, Royal Melbourne Institute of Technology, 1992.

[19.] M. P. Fourman, »Compaction of symbolic layout using genetic algorithms,« u J. J. Grefenstette, editor, Proceedings of the First International Conference on Genetic Algorithms, pages 141-153. Lawrence Erlbaum Associates, 1985.

[20.] S. Bagchi, S. Uckun, Y. Miyabe i K. Kawamura, »Exploring problem specific recombination operators for job shop scheduling,« u R. K. Belew and L. B. Booker, editors, Proceedings of the Fourth International Conference on Genetic Algorithms, pages 10-17. Morgan Kaufman, 1991.

[21.] L. Davis, »Job shop scheduling with genetic algorithms,« u J. J. Grefenstette, editor, Proceedings of the First International Conference on Genetic Algorithms, pages 136-140. Lawrence Erlbaum Associates, 1985.

[22.] G. Syswerda, »Schedule optimization using genetic algorithms,« u Handbook of Genetic Algorithms, Van Nostrand Reinhold, 1991, pp. 332-349.

[23.] D. Whitley, T. Starkweather i D. Fuquay, »Scheduling problems and travelling salesmen: The genetic edge recombination operator,« u J. D. Schaffer, editor, Proceedings of the Third International Conference on Genetic Algorithms, pages 133-140. Morgan Kaufmann, 1989.

[24.] J. R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, 1992.

[25.] J. R. Koza, Genetic Programming II: Automatic Discovery of Reusable Programs, MIT Press, 1994.

[26.] C. De la Higuera, Grammatical Inference: Learning Automata and Grammars, Cambridge: Cambridge University Press, 2010.

[27.] C. De la Higuera, »A bibliographical study of grammatical inference,« Pattern Recognition, svez. 38, pp. 1332-1348, 2005.

[28.] Ž. Kovačević, M. Mernik, M. Ravber i M. Črepinšek, »From Grammar Inference to Semantic Inference—An Evolutionary Approach,« Mathematics, svez. 8, br. 5, 2020.

[29.] M. Ravber, Ž. Kovačević, M. Črepinšek i M. Mernik, »Inferring Absolutely Non-Circular Attribute Grammars with a Memetic Algorithm,« Applied Soft Computing Journal, svez. 100, 2021.

[30.] D. E. Knuth, »Semantics of context-free languages,« Mathematical systems theory, svez. 2, pp. 127-145, 1968.

[31.] P. Moscato i C. Cotta, »A modern introduction to memetic algorithms,« u M. Gendreau, J.-Y. Potvin (Eds.), Handbook of Metaheuristics, Springer US, Boston, MA, pp. 141-183, 2010.

AUTORI · *AUTHORS*

● Vedran Mihelj
Student je preddiplomskog stručnog studija računarstva na Tehničkom veleučilištu u Zagrebu. Slobodno vrijeme posvećuje istraživanju i nadopuni znanja iz područja strojnog učenja i sigurnosti računalnih sustava.

● Aleksandar Stojanović
Predavač je na Tehničkom veleučilištu u Zagrebu i izvodi nastavu na predmetima iz područja programskog inženjerstva, objektno-orijentiranog programiranja i naprednog programiranja. Godine 1996. diplomirao je informacijske i komunikacijske znanosti na Filozofskom fakultetu sveučilišta u Zagrebu, a 1999. godine magistrirao računarstvo u SAD-u na sveučilištu Midwestern State University te je radio kao programski inženjer u području telekomunikacija, financija i energetike. Godine 2019. doktorirao je na Filozofskom fakultetu sveučilišta u Zagrebu s disertacijom pod naslovom "Metoda automatske detekcije naglašenih riječi u zvučnom zapisu". Područja interesa su mu principi i metodologija programiranja te primjena evolucijskog računalstva u rješavanju problema. Autor je knjige “Elementi računalnih programa s primjerima u Pythonu i Scali”.

Korespondencija · *Correspondence*
aleksandar.stojanovic@tvz.hr