PD.TVZ.HR
POLYTECHNIC & DESIGN Vol. 10, No. 4, 2022.
Mrežna inačica: ISSN 2459-6302
Tiskana inačica: ISSN 1849-1995
DOI: 10.19279/TVZ.PD.2022-10-4-05

GENERIRANJE I OPTIMIZACIJA RASPOREDA KORIŠTENJEM GENETSKOG ALGORITMA

GENERATING AND OPTIMIZING SCHEDULES USING GENETIC ALGORITHM

Toni Bedalov[1], Željko Kovačević[2]

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

SAŽETAK

U ovom radu opisan je postupak generiranja i optimizacije rasporeda zaposlenika tvrtke korištenjem genetskog algoritma. Općenito, evolucijsko računanje već se dugo koristi za rješavanja ovakvih tipova problema gdje je potrebno pretraživati velika područja svih mogućih rješenja u najkraćem mogućem roku. Budući da su takvi optimizacijski problemi najčešće kompleksnosti "NP-hard", "brute force" pristup nije primjenjiv jer obično košta previše procesorskog vremena. U rješavanju našeg problema koristili smo sve genetske operatore (selekcija, križanja i mutacija), dok je za potrebe implementacije razvijena aplikacija u programskom jeziku Java. Dobiveni rezultati u gotovo svim slučajevima predstavljaju optimalna rješenja (rasporede), a kada nije moguće doći do optimalnog rješenja, naš pristup daje jedno ili više rješenja koja su najbliža traženom optimumu. Pojedini genetski operatori i dobiveni rezultati opisani su u nastavku.

Ključne riječi: *genetski algoritmi, evolucijsko računanje, optimizacija, raspored*

ABSTRACT

This paper describes the process of generating and optimizing company employee schedules using genetic algorithm. In general, evolutionary computing has long been used to solve these types of problems where it is necessary to search large areas of all possible solutions in the shortest possible time. Since such optimization problems are most often of "NP-hard" complexity, the "brute force" approach is not applicable because it usually costs too much CPU time. In solving our problem, we used all genetic operators (selection, crossover, and mutation), while for the needs of implementation, an application in the Java programming language was developed. The obtained results in almost all cases represent optimal solutions (schedules), and when it is not possible to reach the optimal solution, our approach gives one or more solutions that are closest to the required optimum. The individual genetic operators and the obtained results are described below.

Keywords: *genetic algorithms, evolutionary computing, optimization, schedule*

1. UVOD
1. INTRODUCTION

Raspodjela posla i izrada rasporeda zaposlenika vrlo često je dugotrajan posao. Prilikom raspodjele posla potrebno je pridržavati se svih zakona propisanih unutar Zakona o radu, te je poželjno da taj posao bude ravnomjerno raspodijeljen. U tu svrhu izrađena je aplikacija za generiranje rasporeda zaposlenika korištenjem genetskog algoritma namijenjena tvrtkama u kojima samo jedan zaposlenik radi u smjeni. Evolucijski pristup pokazao se učinkovitim u rješavanju mnogih optimizacijskih problema. Tako je u prošlosti bio korišten prilikom dizajna turbina za avione [1], u odabiru optimalne kombinacije stanica za praćenje rijeke Gediz (Turska) [2], produkcijskom planiranju i optimizaciji narudžbi hrane [3] itd. Pri odabiru

ovog pristupa prethodno smo razmotrili i neke slične probleme koji su uspješno riješeni ovim pristupom poput optimizacije i izrade rasporeda za operacije u Meksičkoj javnoj bolnici [4] te izrade rasporeda za predavanja [5] [6]. Zbog iznimno velikog prostora za pretraživanje (broja svih mogućih kombinacija rasporeda) pristup "brute-force" ne bi bio dovoljno učinkovit u smislu performansi. Međutim, u svim navedenim slučajevima pokazalo se da evolucijski pristup korištenjem genetskog algoritma daje dobra rješenja u vrlo kratkom vremenu, zbog čega smo se odlučili na njegovo korištenje i implementaciju. Konačno, i u slučaju našeg problema ovaj pristup se pokazao učinkovitim, gdje se u obavljenim eksperimenta u relativno brzom vremenu (do nekoliko sekundi) može pronaći optimalno rješenje.

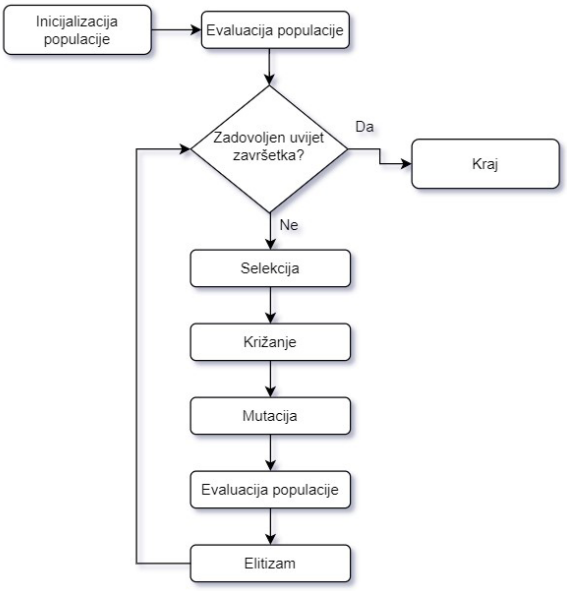
2. PROBLEM RASPOREĐIVANJA
2. SCHEDULING PROBLEM

Problem raspoređivanja posla zaposlenika zahtjeva izradu tjednog rasporeda pri čemu je potrebno slijediti zakone koji definiraju prihvatljiv raspored. Stavkom 3 članka 65. Zakona o radu propisano je da puno radno vrijeme zaposlenika ne smije biti duže od 40 sati tjedno. Ukoliko zaposlenik radi prekovremeno, ukupno tjedno radno vrijeme ne smije biti duže od 50 sati [7]. Uz pridržavanje Zakona definirano je još nekoliko pravila s ciljem omogućavanja odmora i pravednije raspodjele posla među zaposlenicima. Tim pravilima određeno je da ni jedan zaposlenik ne bi trebao raditi dvije smjene uzastopno ili u istom danu.

Prostor pretraživanja, odnosno broj svih mogućih potencijalnih rješenja u ovom slučaju iznosi $z^{(d*s)}$ gdje z predstavlja broj zaposlenika, d broj radnih dana u tjednu, a s broj smjena u pojedinom danu. Primjerice, ako tvrtka ima potrebu za raspoređivanjem posla između 3 radnika u 3 smjene tijekom 5 radnih dana, tada postoji preko 14 milijuna mogućih rasporeda od kojih samo nekoliko može poslužiti kao prihvatljivo rješenje. S obzirom na veličinu takvog prostora pretraživanja, pronalazak rješenja pristupom kao što je „brute force“ zahtijevao bi previše vremena. Genetski algoritam opisan u ovom radu pokazao se iznimno učinkovitim za rješavanje tog problema jer, ovisno o korištenim postavkama, najčešće pronalazi rješenje unutar svega 20 iteracija (generacija).

3. GENETSKI ALGORITAM
3. GENETIC ALGORITHM

Genetski algoritam (GA) metoda je pretraživanja koja oponaša prirodni evolucijski proces kako bi se pronašlo optimalno rješenje među velikim brojem potencijalnih rješenja [8] [9] [10] [11]. GA započinje generiranjem populacije koja predstavlja skup jedinki, a svaka jedinka predstavlja jedno potencijalno rješenje. Svojstva jedinke definirana su genima, a skup svih gena jedinke naziva se genotip. Iako većina genetskih algoritama implementira genotip kao niz bitova, često su poželjne drugačije reprezentacije koje bolje prikazuju sadržane podatke [12]. U izrađenom GA genotip je implementiran kao dvodimenzionalni niz cijelih brojeva, a svaki gen unutar genotipa predstavlja jednu smjenu u rasporedu. Vrijednost unutar gena može biti bilo koji ID iz liste aktivnih zaposlenika unesenih u bazu podataka.



Slika 1 *Dijagram toka genetskog algoritma*

Figure 1 *Flow diagram of a genetic algorithm*

Kao što to pokazuje Slika 1, evolucijski proces započinje inicijalizacijom, odnosno generiranjem slučajne populacije. S obzirom na ogromnu

veličinu prostora za pretraživanje svih mogućih rješenja malo je vjerojatno da se optimalno rješenje nalazi upravo u inicijalnoj (slučajno generiranoj) populaciji. Zato se nakon njene evaluacije najčešće ulazi u petlju selekcije, križanja i mutacije. Nakon evaluacije svake nove generacije kao zadnji korak izvršava se postupak elitizma (očuvanje najkvalitetnijih pojedinaca iz prethodne generacije) te se petlja ponavlja sve dok se ne dođe do optimalnog rješenja ili korisničkog prekida.

3.1. FUNKCIJA DOBROTE
3.1. FITNESS FUNCTION

Funkcija dobrote (eng. fitness function) služi za određivanje dobrote (eng. fitness), odnosno kvalitete pojedine jedinke ovisno o njenom genotipu kako bi se utvrdilo koje su jedinke najpodobnije za sudjelovanje u idućoj generaciji [13]. Ona mora biti definirana tako da vjerno odražava problem koji se rješava [14]. U konvencionalnom GA najbolja rješenja imaju najveći iznos dobrote, dok najgora rješenja imaju najmanji iznos.

Kako problem izrade rasporeda u našem slučaju zahtjeva slijeđenje određenih pravila propisanih Zakonom, zbog jednostavnosti implementacije računa se kazna jedinke za kršenje svakog od zadanih pravila. U tom slučaju optimalan raspored ima vrijednost dobrote 0, što znači da ne krši ni jedno pravilo, a lošija rješenja imaju negativne vrijednosti. Pod optimalnim rasporedom smatramo rješenja u kojima ni jedan zaposlenik ne radi prekovremeno, više smjena uzastopno ili u istom danu.

Iako nepoželjna, rješenja s negativnom dobrotom nisu nužno neprihvatljiva. Nekada broj zaposlenika nije dovoljan da bi se izbjegao prekovremeni rad te u takvim situacijama nije moguće generirati optimalan raspored. Zbog toga algoritam, ukoliko je to moguće, korištenjem funkcije dobrote raspoređuje prekovremene sate tako da ni jedan od zaposlenika ne radi preko 50 sati tjedno. Takav način pretraživanja postiže se tako što se svaki sat iznad normalnog radnog vremena (40 sati) kažnjava relativno malim iznosom kazne. Rješenja u kojima zaposlenik radi preko 50 sati također se kažnjavaju, ali znatno većim iznosom.

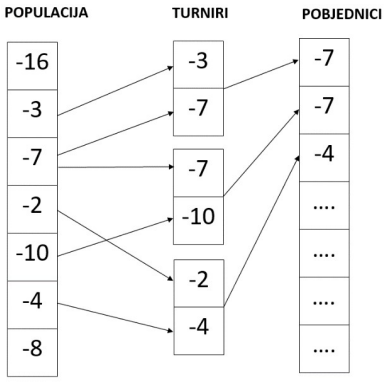
Pravila i vrijedosti kazne koja se dodjeljuju za njihova kršenja su sljedeća:

1. Zaposlenik radi više od jedne smjene u istom danu → **- 30**
2. Zaposlenik radi zadnju smjenu i prvu smjenu idući dan → **-20**
3. Zaposlenik radi više od 50 sati u tjednu → **Broj radnih sati iznad 50 * (-6)**
4. Zaposlenik radi više od 40 sati u tjednu → **Broj radnih sati iznad 40 * (-2)**

Važno je napomenuti da svako od navedenih pravila može biti prekršeno više puta unutar jednog rješenja i da se svako takvo kršenje iznova kažnjava zadanom vrijednošću kazne.

3.2. SELEKCIJA
3.2. SELECTION

Nakon pokretanja algoritma i generiranja populacije slijedi operator selekcije. Selekcija je proces odabira najboljih jedinki iz generacije na temelju iznosa dobrote kako bi njihovi genomi sudjelovali u kreiranju nove generacije [15]. Postoji više vrsta selekcije (primjerice, AB-selekcija, adaptivna, spolna selekcija itd.), a za ovu implementaciju odabrana je turnirska selekcija. Ona radi na načelu turnira, tako što se n jedinki natječe, a pobjeđuje ona koja ima najbolju dobrotu. Razvijena aplikacija koristi selekciju s 2 natjecatelja s pretpostavkom da bi veći broj natjecatelja rezultirao velikim brojem duplikata, što bi smanjilo raznolikost populacije. Turniri se ponavljaju dok god nije odabran broj jedinki jednak veličini ukupne populacije, pri čemu ista jedinka može biti odabrana više puta.



Slika 2 *Turnirska selekcija*

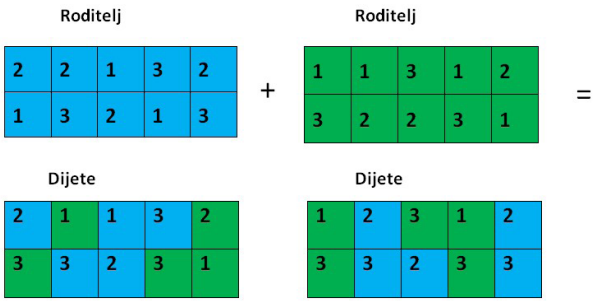
Figure 2 *Tournament selection*

Slika 2 prikazuje postupak turnirske selekcije. Iz slike je vidljivo da se iz populacije odabiru dvije nasumične jedinke koje zatim sudjeluju u turniru. Pobjednik turnira je jedinka s većim iznosom dobrote, odnosno manjim iznosom kazne, te ona postaje dio nove populacije. Turniri se ponavljaju dok broj odabranih jedinki nije jednak ukupnom broju jedinki iz prethodne populacije. Na primjeru jedinke iz slike čiji iznos kazne iznosi -7 također je vidljivo da ista jedinka može sudjelovati i pobijediti u više turnira.

3.3. KRIŽANJE
3.3. CROSSOVER

Križanje (eng. crossover) postupak je kombiniranja genoma roditeljskih jedinki kako bi se proizvela jedna ili više jedinki djece koja će biti dio iduće generacije. Dobro osmišljen operator križanja može uvelike poboljšati učinkovitost pretrage [16]. Jedinke koje su prošle selekciju kandidati su za križanje, a vjerojatnost njihova odabira u našim je eksperimentima najčešće iznosila 60%. Ako je broj jedinki odabranih za križanje neparan zadnja dodana jedinka se uklanja iz liste kako bi se omogućilo da sve odabrane jedinke, odnosno roditelji, mogu biti uparene.

Nakon odabira jedinki za križanje nasumično se odabiru parovi roditelja. Prilikom križanja nastaju dvije jedinke (djeca) čiji su geni s jednakom vjerojatnošću (50%/50%) naslijeđeni od jedne ili druge roditeljske jedinke. Novonastale jedinke tada mijenjaju svoje roditelje u populaciji. U procesu križanja svaki roditelj može sudjelovati samo jedanput.



Slika 3 *Križanje*

Figure 3 *Crossover*

Slika 3 prikazuje postupak križanja dvaju roditelja te dviju novonastalih jedinki djece kao produkt. Geni djece prikazani plavom bojom naslijeđeni su od prvog roditelja, dok su geni prikazani zelenom bojom naslijeđeni od drugog roditelja. Kao što je već spomenuto, svaki gen unutar genoma predstavlja pojedinu smjenu, dok vrijednosti unutar gena predstavljaju ID zaposlenika zaduženog za tu smjenu.

3.4. MUTACIJA
3.4. MUTATION

Operator mutacije (eng. mutation) koristi se nakon operatora križanja kako bi se očuvala genetska raznolikost populacije među generacijama [17]. U slučaju zapinjanja algoritma na lokalni prostor pretrage (primjerice, zbog lokalnog optimuma) mutacija povećava vjerojatnost prelaska u globalni prostor te samim time i vjerojatnost pronalaska globalnog optimuma koji predstavlja traženo rješenje [18]. Lokalni optimum se događa zbog nemogućnosti optimizacijskog algoritma da nađe bolje rješenje u nekom lokalnom području, a istovremeno i zbog nemogućnosti prelaska pretrage u druge lokalne prostore (npr. zbog niske vjerojatnosti mutacije). Tada se događa beskonačna petlja u kojoj nije moguće pronaći globalni optimum. Mutacija je implementirana tako da svaki gen unutar populacije u svakoj iteraciji algoritma ima vjerojatnost da promjeni svoju vrijednost u bilo koji ID iz liste aktivnih zaposlenika. Vjerojatnost mutacije mora biti relativno mala jer u suprotnom algoritam prelazi u nasumično pretraživanje te zbog toga nismo koristili vjerojatnost mutacije veću od 10%.

Algoritam 1

Algorithm 1

Za svaku jedinku **u** populaciji
  Za svaki gen **u** genotipu jedinke
    Generiraj nasumični broj **x** od 0 do 1
    Ako je **x** <= vjerojatnost mutacije **onda**
      Generiraj nasumični broj **y** od 0 do (veličina populacije – 1)
      gen = ID zaposlenika s indexom **y**

Algoritam 1 opisuje način na koji se implementira mutacija pojedinog gena. Svaki gen unutar populacije ima zadanu vjerojatnost da će biti mutiran, te je zbog toga za sve gene potrebno generirati nasumični broj u vrijednosti od 0 do

1. Ukoliko je generirani broj manji od zadanog postotka vjerojatnosti mutacije u decimalnom obliku tada se vrijednost, odnosno zaposlenik sadržan unutar tog gena, zamjenjuje s nekim drugim nasumičnim zaposlenikom.

3.5. ELITIZAM
3.5. ELITISM

Zadnji korak iteracije genetskog algoritma je elitizam (eng. elitism), a njegova uloga je očuvanje najboljih jedinki kako bi se njihovi geni prenijeli u iduću generaciju. Elitizam znatno ubrzava rad genetskog algoritma i sprječava gubitak dobrih rješenja nakon njihovog pronalaska [19]. Zahvaljujući elitizmu najbolja jedinka iz trenutne generacije ne može biti gora od najbolje jedinke iz prošle generacije te zbog toga algoritam prilikom svake iteracije teži pronalasku boljeg rješenja.

Elitizam se postiže tako što se na kraju svake iteracije **n** najlošijih jedinki trenutne generacije zamjenjuje s **n** najlošijih jedinki iz prethodne generacije. Nakon operatora mutacije potrebno je evaluirati populaciju kako bi se mogao primijeniti elitizam i provjeriti je li pronađeno traženo rješenje. Broj elitnih jedinki određen je postotkom u odnosu na ukupnu veličinu populacije, a najčešće korištena vrijednost ovog parametra iznosila je 10%.

3.6. REZULTATI
3.6. RESULTS

Pretpostavimo da nam je zadatak pronaći optimalni raspored tijekom radnih dana u tjednu (ponedjeljak-petak), gdje se svaki dan radi u 3 smjene (3 zaposlenika) po 8 sati. Korištene su sljedeće postavke genetskog algoritma:

Veličina populacije: 500

Broj natjecatelja (turnirska selekcija): 2

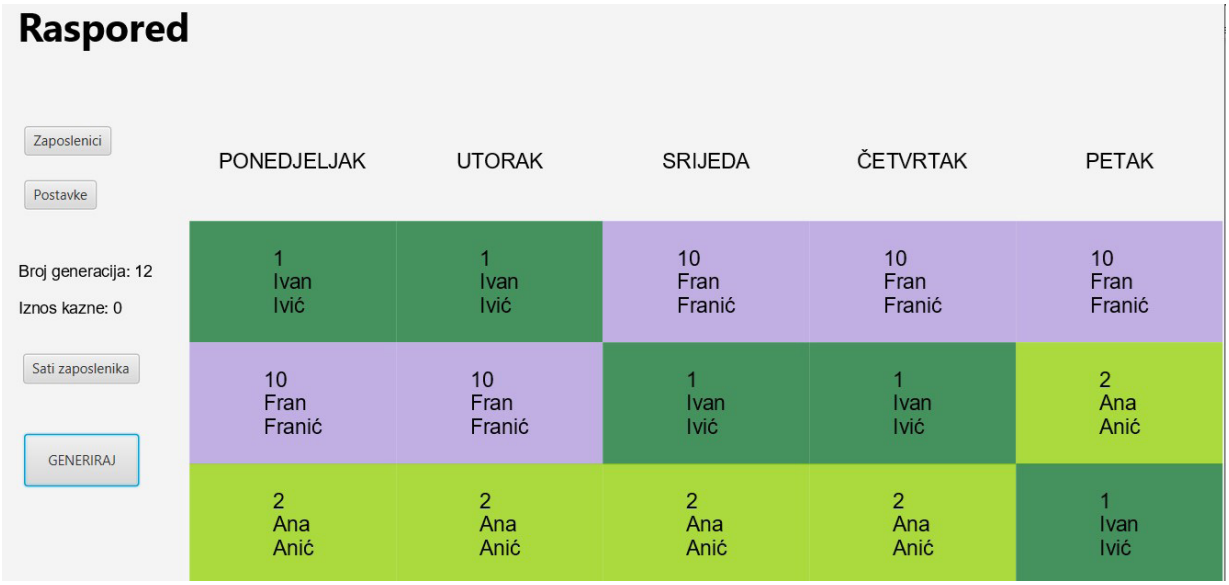
Vjerojatnost križanja: 60%

Vjerojatnost mutacija: 10%

Elitizam: 10%

Kao rješenje dobili smo raspored prikazan na sljedećoj slici (Slika 4).

Rješenje je nastalo kroz 12 iteracija (generacija) te je ukupno vrijeme pretrage prostora iznosilo tek 73 milisekunde. Na slici je vidljivo da svaki zaposlenik radi 5 dana, a s obzirom da duljina smjene definirana u postavkama iznosi 8 sati, svaki zaposlenik radi 40 sati tjedno što znači da nitko od njih ne radi prekovremeno. Uz to, niti jedan od zaposlenika ne radi više od jedne smjene uzastopno kao niti više od jedne smjene u istom danu. Takav raspored ne krši ni jedno od definiranih pravila te je zbog toga iznos kazne 0.



Slika 4 *Prikaz generiranog rješenja*

Figure 4 *Representation of a generated solution*

4. ZAKLJUČAK
4. CONCLUSION

Uz sve prethodno riješene probleme sličnog tipa, i ovaj rad dodatno potvrđuje da je evolucijsko računanje zaista odgovarajući pristup za pretraživanje velikih prostora svih mogućih rješenja. Ipak, unatoč dobrim rezultatima i uspješnom pronalasku optimalnih rasporeda, postoje razne mogućnosti za daljnje poboljšanje. Tako kanimo razmotriti mogućnost dodatnog ubrzanja procesa pretrage implementacijom jedne ili više strategija lokalnog pretraživanja (memetski algoritam [20]) te korištenje tablice raspršivanja u svrhu sprečavanja ponovnih evaluacija istih pojedinaca. Na taj način vjerujemo da bi prethodno opisani pristup imao mnogo bolje performanse pri rješavanju istih problema, ali ovaj puta i s puno većim brojem ulaznih parametara, odnosno prilikom pretraživanja i mnogo većih prostora mogućih rješenja.

5. REFERENCE
5. REFERENCES

[1.] A. I. Belousov and A. Y. Sapozhnikov, "Synthesis of basic structural design of aircraft GTE based on genetic algorithms", Russ. Aeronaut. , vol. 58, pp. 199-204, 2015., doi: https://doi.org/10.3103/S1068799815020105

[2.] Y. Icaga, "Genetic Algorithm Usage in Water Quality Monitoring Networks Optimization in Gediz (Turkey) River Basin", Environmental Monitoring and Assessment, vol. 108, p. 261–277, 2005., doi: https://doi.org/10.1007/s10661-005-4328-z

[3.] T. Brajković, M. Perinić and M. Ikonić, "Production Planning and Optimization of Work Launch Orders Using Genetic Algorithm", Tehnički vjesnik, vol. 25, no. 5, pp. 1278-1285, 2018., doi: https://doi.org/10.17559/TV-20161207195125

[4.] G. L. C. P. S.-S. N. R.-V. a. J. R.-O. Rivera, "Genetic Algorithm for Scheduling Optimization Considering Heterogeneous Containers: A Real-World Case Study", Axioms , 2009., doi: https://doi.org/10.3390/axioms9010027

[5.] R. H. David Kristiadi, "Genetic Algorithm for Lecturing Schedule Optimization (Case Study: University of Boyolali)", IJCCS (Indonesian Journal of Computing and Cybernetics Systems), vol. 13, no. 1, 1 2019., doi: 10.22146/ijccs.43038

[6.] J. Xu, "Improved Genetic Algorithm to Solve the Scheduling Problem of College English Courses", Complexity, 2021., doi: https://doi.org/10.1155/2021/7252719

[7.] "Zakon o radu RH", 2020. [Online]. Available: https://www.zakon.hr/z/307/Zakon-o-radu. [Accessed 22. kolovoza 2021.].

[8.] T. Manning, R. D. Sleator and P. Walsh, "Naturally selecting solutions - The use of genetic algorithms in bioinformatics", 2012. [Online]., doi:10.4161/bioe.23041, Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3813526/. [Accessed 22. kolovoza 2021.].

[9.] J. R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems), Cambridge, Massachusetts, USA: The MIT Press, 1992., ISBN: 978-0262111706

[10.] A. Lambora, K. Gupta and K. Chopra, "Genetic Algorithm- A Literature Review", 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), Faridabad, India, IEEE, DOI: 10.1109/COMITCon.2019.8862255

[11.] T. Alam, S. Qamar, A. Dixit and M. Benaida, "Genetic Algorithm: Reviews, Implementations, and Applications", International Journal of Engineering Pedagogy (iJEP), 2020, https://doi.org/10.48550/arXiv.2007.12673

[12.] J. S. Dean, Staff Scheduling by a Genetic Algorithm with a Two-Dimensional Chromosome Structure, 2012.

[13.] G. Bakırlı, D. Birant and A. Kut, "An incremental genetic algorithm for classification and sensitivity analysis of its parameters", in Expert Systems With Applications, 2011., doi: http://dx.doi.org/10.1016/j.eswa.2010.08.051

[14.] M. Golub, "Skripta 1. dio : Genetski algoritmi", 27. rujan 2004. [Online].

Available: http://www.zemris.fer.hr/~golub/ga/ga_skripta1.pdf. [Accessed 22. kolovoza 2021.].

[15.] N. M. Razali and J. Geraghty, "Genetic Algorithm Performance with Different Selection Strategies in Solving TSP", 2011.

[16.] D. Sudhold, "How crossover speeds up building-block assembly in Genetic Algorithms", 2012., doi: 10.1162/EVCO_a_00171

[17.] M.-W. Tsai, T.-P. Hong and W.-T. Lin, "A Two-Dimensional Genetic Algorithm and Its Application to Aircraft Scheduling Problem", in Mathematical Problems in Engineering, 2015., doi: https://doi.org/10.1155/2015/906305

[18.] V. Bedek, "Stvaranje rasporeda sati genetskim algoritmima", 2008., završni rad, Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu, Hrvatska

[19.] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," in IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, 2002., doi: 10.1109/4235.996017

[20.] P. Moscato, "On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts - Towards Memetic Algorithms. Caltech Concurrent Computation Program," 2000.

AUTORI · *AUTHORS*

• **Toni Bedalov** - Student je preddiplomskog stručnog studija računarstva na Tehničkom veleučilištu u Zagrebu. Izradio je aplikaciju za vođenje evidencije članova sportskih klubova, te je sudjelovao u projektima izrade web aplikacija korištenjem Spring programskog okvira i MySQL baze podataka. U slobodno vrijeme se aktivno bavi Judom i unaprjeđivanju znanja iz područja objektno orijentiranih programskih jezika.

• **Željko Kovačević** - Viši je predavač na Tehničkom veleučilištu u Zagrebu gdje sudjeluje u nastavi iz kolegija orijentiranih prema učenju programskih jezika i baza podataka. Doktorirao je na Fakultetu elektrotehnike, računarstva i informatike u Mariboru 2022. godine na području razvoja domenski specifičnih programskih jezika. Autor i koautor je 6 knjiga te mnogobrojnih stručnih i znanstvenih radova objavljenih u domaćim i inozemnim časopisima, a 2015.g. dobiva posebno priznanje MVP (Most Valuable Professional) tvrtke Embaracdero za razvoj aplikacija u Embarcadero RAD Studio alatima C++ Builder i Delphi.

Korespondencija · *Correspondence*
zeljko.kovacevic@tvz.hr