Reševanje industrijskih problemov z uporabo računske inteligence: primer avtomatskega načrtovanja delovnega časa

Grega Vrbančič, Iztok Fister ml., Vili Podgorelec

Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, Maribor, Slovenija
grega.vrbancic@um.si, iztok.fister1@um.si, vili.podgorelec@um.si

Sinopsis Dostopnost velikih količin podatkov in relativno poceni in dostopne računske moči je v zadnjih nekaj letih pripomogla k enormnemu vzponu računske inteligence. Čeprav se mnoga podjetja in organizacije že dolga leta poslužujejo uporabe različnih tehnik matematične optimizacije, ki se najpogosteje uporabljajo za namen optimizacije poslovnih procesov, proizvodnih procesov, logistike in upravljanja distribucijskih omrežij, pa so novodobna gigantska podjetja kot so Uber, Netflix in Tesla vpeljala metode in tehnike računske inteligence tudi v bolj uporabniku usmerjene funkcionalnosti ter tako še dodatno povečale njihovo prepoznavnost. V industrijskih okoljih je tako sedaj, bolj kot kadarkoli, pomembno da je organizacija in izraba razpoložljivih virov kar se da učinkovita, saj lahko tako dosežejo pomembno konkurenčno prednost. Pri vpeljavi računske inteligence za reševanje industrijskih problemov pa se podjetja pogosto soočijo s številnimi izzivi, med katerimi pogosto izstopa predvsem zahtevna prilagoditev naprednih optimizacijskih algoritmov, predstavljenih v literaturi, na domensko specifični problem.

V prispevku predstavimo pristope, metode in tehnike, s katerimi lahko učinkovito naslavljamo najpogostejše industrijske probleme, obenem pa opozorimo na morebitne pasti, s katerimi se lahko soočimo pri vpeljavi ter podali smernice za njihovo naslovitev. V nadaljevanju podrobneje predstavimo primer uporabe optimizacijskih algoritmov, ki delujejo po vzorih iz narave, za reševanje problema avtomatske izdelave kompleksnih urnikov.

Ključne besede:
reševanje industrijskih problemov
računska inteligenca
diferencialna evolucija
optimizacija z rojem delcev
avtomatsko načrtovanje

Univerzitetna založba Univerze v Mariboru
ISBN 978-961-286-639-6
DOI https://doi.org/10.18690/um.feri.10.2022.21

1 Uvod

Dandanes je tekmovalnost na globalnem trgu prisotna že od začetka razvoja nekega izdelka. Uspeti z nekim novim produktom, storitvijo ali izdelkom na globalnem trgu ni več samoumevno, saj konkurenca, ki jo najdemo domala na vseh kontinentih, lahko prav tako razvija podoben produkt, ki je lahko celo boljši in cenejši. Zaradi teh razlogov podjetja svoje izdelke poskušajo venomer bolj razvijati in oblikovati na stroškovno učinkovitejši in trajnostni način, da lahko povečajo svoj dobiček in uspešnost ter seveda čim bolj zmanjšajo stroške in potencialne napake. To od inženirjev zahteva, da sledijo najnovejšim trendom v raziskavah ter da uporabljajo stroškovno najučinkovitejše tehnologije in moderne računalniške modele. V zadnjih letih postaja umetna inteligenca ena najbolj priljubljenih orodij v industriji, saj lahko pomaga pri različnih fazah razvoja nekega izdelka ali storitve. Z uporabo umetne inteligence lahko načrtujemo, oblikujemo in izdelujemo izdelke na hitrejši in učinkovitejši način, kjer prav tako že med razvojem poskušamo minimizirati prisotnost napak.

Celotno področje umetne inteligence je izjemno široko saj so raziskovalci že razvili ogromno različnih metod, ki jih lahko štejemo pod okrilje umetne inteligence. Posledično je zato dosti težje odkriti neko metodo, ki je primerna za reševanje določenega domensko specifičnega problema. V tem članku se osredotočamo na računsko inteligenco, ki je podveja umetne inteligence. Računska inteligenca se je v zadnjem desetletju izkazalo kot hitro rastoče področje. Poleg metod in tehnik kot so genetski algoritmi, evolucijsko računalništvo in strojno učenje, računska inteligenca vključuje tudi metode, ki delujejo po vzorih iz narave in poskušajo reševati težke probleme s posnemanjem naravnih sistemov. Računska inteligenca tako predstavlja skupek metod, tehnik in orodij za inteligentno obdelavo podatkov na katerih temelji odločanje in upravljanje v različnih organizacijah od proizvodnih podjetij do medicinskih in zdravstvenih ustanov.

Ne glede na uspešnost uporabe omenjenih metod za reševanje različnih domensko specifičnih problemov v raznovrstnih organizacijah, pa uporaba oz. vpeljava teh prinaša nemalo izzivov s katerimi se moramo soočiti v kolikor želimo prednosti posameznih metod in tehnik računske inteligence kar se da dobro izkoristiti. Nenazadnje lahko že sama izbira metode ali tehnike za naslovitev specifičnega problema predstavlja velik izziv, še posebej v današnjih časih, ko se novi pristopi in rešitve problemov razvijajo na skorajda dnevni ravni. Izbira primerne metode ali tehnike je ključna pri naslovitvi problema, saj lahko z neprimerno izbiro navidezno uspešno rešimo ciljni problem, dolgoročno pa se izkaže da takšna rešitev ne naslavlja problema na način kot smo si ga zadali oz. ne dosega željene uspešnosti. Z neprimerno izbiro metode ali tehnike želenih se lahko dogodi tudi, da ciljev sploh ne dosežemo in posledično opustimo idejo o vpeljavi računske inteligence za rešitev specifičnega domenskega problema ter tako potencialno ne pridobimo konkurenčne prednosti. Poleg same izbire metode ali tehnike pa ne smemo zanemariti tudi problematike same vpeljave te za naslovitev domensko specifičnega problema. Za uspešno vpeljavo moramo namreč dovolj dobro poznati sam problem, podatke, ki so nam na voljo ter določiti omejitve, da lahko oblikujemo model problema, ki zajema vse zahteve in omejitve ter v končni fazi uspešno rešuje zadan problem. Našteti izzivi pa nemalokrat povzročajo probleme tudi ekspertom s področja računske inteligence.

S ciljem približati in olajšati vpeljavo računske inteligence za reševanje realnih industrijskih problemov, v prispevku predstavimo metode in tehnike, s katerimi lahko učinkovito naslavljamo najpogostejše industrijske probleme, obenem pa tudi opozorimo na morebitne pasti, s katerimi se lahko soočimo pri vpeljavi ter podamo smernice za njihovo naslovitev. V nadaljevanju prispevka podrobneje predstavimo primer uporabe dveh metod iz skupine računske inteligence, tj. evolucijskim algoritmom in algoritmom inteligence rojev, za namen reševanja realnega problema avtomatskega načrtovanja delovnega časa. Obe izmed uporabljenih metod sta primerni za reševanje težkih optimizacijskih problemov v raznovrstnih industrijskih okoljih.

2 Uporaba računske inteligence

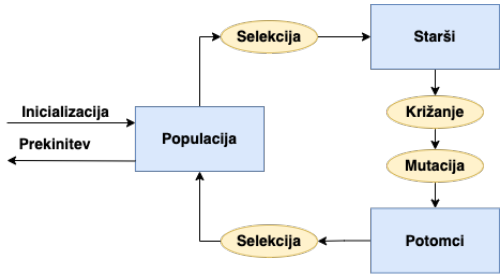
V tem poglavju na kratko opredelimo pojem računske inteligence in predstavimo glavne metode, ki spadajo pod to okrilje. Vsako metodo tudi na kratko opišemo. V nadaljevanju bralca bolj podrobno seznanimo z evolucijskimi algoritmi in algoritmi inteligence rojev, ki so pomembni za nadaljno razumevanje tematike. Poglavje zaključimo s kratkim opisom industrijskih aplikacij, ki temeljijo na uporabi evolucijskih algoritmov in algoritmov inteligence rojev.

2.1 Računska inteligenca

Računska inteligenca (angl. Computational Intelligence, krajše CI) obsega metode, ki delujejo po vzorih iz narave in poskušajo reševati težke probleme s posnemanjem principov naravnih sistemov [1]. V skupino metod računske inteligence štejemo: umetne nevronske mreže (angl. Artificial Neural Networks, krajše ANN) [4], umetne imunske sisteme (angl. Artificial Immune Systems, krajše AIS) [6], sisteme na osnovi mehke logike (angl. Fuzzy Systems, krajše FS) [3], evolucijske algoritme (angl. Evolutionary Algorithms, krajše EA) [5], ter algoritme inteligence rojev (angl. Swarm Intelligence, krajše SI) [2]. Umetne nevronske mreže delujejo po vzgledu delovanja človeških možganov in se precej uporabljajo za probleme, ki vključujejo klasifikacijo, razpoznavanje vzorcev, regresijske probleme kot tudi za gradnjo napovednih modelov. Umetni imunski sistemi temeljijo na principu delovanja naravnih imunskih sistemov in njihova glavna značilnost je sposobnost razpoznavanja vzorcev, zato veliko aplikacij uporabe umetnih imunskih sistemov najdemo na področju varnosti računalniških sistemov in iskanja anomalij. Prav tako so tudi odlično orodje za reševanje optimizacijskih problemov. Sistemi mehke logike uporabljajo aproksimativno sklepanje namesto eksaktnega sklepanja in se posledično zato dobro obnesejo pri problemih regulacije, nadzoru robotskih sistemov, ipd. Evolucijski algoritmi posnemajo Darwinovo evolucijsko teorijo in so zelo učinkovito orodje za reševanje težkih optimizacijskih problemov v računalništvu, fiziki, matematiki, medicini in tudi v industriji. Sorodna skupina evolucijskih algoritmov so algoritmi inteligence rojev, ki temeljijo na agentih, kjer je vsak agent sposoben izvajanja enostavnega opravila. V primeru povezave v skupnost so sposobni tudi kompleksnega obnašanja [1].

2.2 Evolucijski algoritmi in algoritmi inteligence rojev

Evolucijske algoritme in algoritme inteligence rojev štejemo tudi v skupino stohastičnih populacijskih algoritmov po vzorih iz narave (angl. Stochastic population-based nature-inspired algorithms). Njihovo delovanje je navdihnjeno s strani narave in različnih procesov, ki potekajo v naravi [7]. Prvi predstavniki, ki so se pojavili že v zgodnjih 60-tih letih prejšnjega stoletja so evolucijski algoritmi. Njihov glavni navdih je Darwinova evolucijska teorija, ki pravi, da imajo posamezniki, ki se najbolj prilagodijo razmeram v okolju več možnosti, da preživijo in se posledično reproducirajo. Slabše posameznike pa proces naravne selekcije izloči iz populacije. Osnovni evolucijski algoritem sestoji iz populacije posameznikov, ki je podvržena različnim variacijskim operatorjem, kot npr. mutacija, križanje in selekcija. Slika 1 prikazuje osnoven princip delovanja splošnega evolucijskega algoritma.



Slika 1: Osnoven princip delovanja evolucijskih algoritmov.
Vir: lasten.

Skozi obdobje razvoja evolucijskih algoritmov se je pojavilo več različnih variacij teh algoritmov. Dandanes družina evolucijskih algoritmov zajema: genetske algoritme, evolucijske strategije, genetsko programiranje in diferencialno evolucijo (DE). Glavne razlike med temi skupinami ležijo v predstavitvi posameznikov v populaciji, npr. pri genetskih algoritmih so posamezniki predstavljeni z binarnim zapisom, medtem ko so v diferencialni evoluciji posamezniki predstavljeni z vektorjem realnih števil. Poleg predstavitve posameznikov v populaciji so glavne razlike še v uporabi variacijskih operatorjev.

Druga skupina so algoritmi inteligence rojev, kateri so navdihnjeni na podlagi socializiranih žuželk (npr. mravelj, čebel, ipd.) in nekaterih vrst socializiranih živali (npr. ptic, rib, ipd.). Te živali so sposobne opravljanja kompleksnih opravil v skupini, čeprav jim je narava kot posameznikom namenila avtonomno izvajanje samo preprostih akcij. Definicija iz članka [8] pravi, da se inteligenca rojev nanaša na kolektivno nastajajoče vedenje več agentov, ki pri sobivanju upoštevajo nekaj preprostih pravil. Medtem ko lahko agenta kot posameznika obravnavamo kot omejenega, ker je sposoben izvajanja zgolj preprostih akcij, pa večagentni sistem, kjer ti delujejo kot celota, lahko kaže samoorganizacijsko vedenje in izkazujejo določeno vrsto kolektivne inteligence.

2.3 Reševanje industrijskih problemov

Področje evolucijskih algoritmov in algoritmov inteligence rojev je izredno dinamično, saj se skoraj vsak dan pojavijo kakšni novi oziroma prilagojeni algoritmi, ki so primerni za reševanje praktičnih problemov. Domala skoraj ni možno najti področja, kjer ni vsaj ene aplikacije, ki temelji na teh pristopih. Nekaj izbranih primerov je prikazanih v nadaljevanju.

Evolucijski algoritmi se uporabljajo za načrtovanje in optimizacijo različnih anten [10], avtomatski razvoj avtomobilskih motorjev [12], optimizacijo poti pri brezpilotnih vozilih [13]. Njihova prisotnost je utrjena tudi na področju energetike, kjer se uporabljajo za ocenjevanje parametrov fotovoltaičnih modelov [14], iskanja najboljših pozicij za postavitev vetrnih elektrarn [15] in napovedovanja proizvodnje električne energije [16, 17]. Algoritmi inteligence rojev se prav tako uporabljajo za reševanje podobnih problemov, vendar se veliko teh pristopov uporablja tudi za reševanje kombinatoričnih optimizacijskih problemov [21, 22]. Primeri uporabe obeh skupin pristopov se lahko najdejo tudi na ostalih področjih, kot so področje športa [18], ekonomije in bančništva [11] ter medicine [19]. Za več primerov uporabe vabimo bralca k branju naslednjega prispevka [20].

3 Vpeljava računske inteligence za naslovitev industrijskih problemov

V prispevku smo se omejili na vpeljavo algoritmov iz skupin evolucijskih algoritmov in algoritmov inteligence rojev. Kot smo predstavili v prejšnji sekciji so algoritmi iz teh skupin uspešno uporabljeni za naslovitev številnih industrijskih problemov pri čemer dosegajo visoko stopnjo uspešnosti. Za uspešno vpeljavo tovrstnih algoritmov je potrebno razumeti njihov osnoven princip delovanja. Razumevanje osnovnega principa delovanja je namreč ključno, da lahko ciljni problem v osnovi ustrezno opredelimo.

3.1 Kdaj uporabiti evolucijske algoritme ali algoritme inteligence rojev?

V praksi se velikokrat srečamo z novim problemov za katerega potrebujemo rešitev. Za veliko problemov obstaja kopica rešitev in jih lahko zelo hitro rešimo oziroma se približamo željeni rešitvi. Včasih pa za določen problem ni na voljo neke metode s pomočjo katere pridemo do rešitve. Včasih smo prav tako postavljeni pred nek problem za katerega nikakor ne moremo izpeljati neke metode, ki bi nas privedla do rešitve. Prav tako veliko ljudi v industrijskih okoljih ne zanima kako smo prišli do rešitve, samo da jo najdemo [30]. Če ima naš problem podobne karakteristike, potem je velika možnost, da se ga da rešiti z metodami računske inteligence oziroma bolj natančno z evolucijskimi algoritmi ali algoritmi inteligence rojev. A po drugi strani je zelo pomembno še omeniti, da so določeni problemi, ki jih rešujemo z eksaktnimi metodami zelo časovno in prostorsko zahtevni. V tem primeru

nam lahko uporaba metod računske inteligence eliminira veliko časovno zahtevnost, vendar nam ne zagotovi najbolj optimalne rešitve.

3.2 Opredelitev in modeliranje problema

Preden začnemo načrtovati evolucijski algoritem za določen problem je potrebno problem natančno opredeliti. Takoj na začetku je potrebno določiti kakšen izhod pričakujemo in če obstajajo kakšni vhodi. Opredeliti je potrebno tudi vse omejitve in predpostavke za določen problem. Zatem je potrebno določiti kaj nam bodo predstavljali posamezniki oziroma kandidatne rešitve v populaciji v evolucijskem ciklu. Če iščemo npr. najkrajšo pot od točke A do točke B, potem lahko posameznik predstavlja vozlišča med obema točkama. V primeru da iščemo optimalni načrt športnega treninga, potem nam lahko posamezniki predstavljajo obremenitev treninga po dnevih. Na podlagi tega je potem lažje izbrati tudi primeren algoritem. Če so posamezniki predstavljeni kot zaporedje binarnih števil, potem je smiselno izbrati genetski algoritem, če so posamezniki predstavljeni kot programi, potem se lahko zatečemo k genetskemu programiranju. Če so posamezniki predstavljeni kot vektorji realnih števil, potem lahko uporabimo na primer diferencialno evolucijo. Po tem koraku je potrebno definirati kriterijsko funkcijo. Naloga kriterijske funkcije je ocenitev kvalitete posameznika v populaciji. Na podlagi te ocenitve se potem boljši posamezniki prenesejo v naslednjo generacijo, medtem ko se slabši posamezniki ponavadi zavržejo oziroma je to čisto odvisno od algoritma. Po definiciji kriterijske funkcije pa je potrebno še potem tudi prilagoditi variacijske operatorje in/ali dodati kakšno lokalno iskanje, s katerimi lahko izboljšamo delovanje algoritma za nek specifičen nabor ciljnih problemov.

3.3 Izzivi

Trenutno obstaja izjemno veliko število različnih algoritmov, ki jih je predlagala akademska skupnost [31]. V vsej tej poplavi je zelo težko za začetnika izbrati najbolj primeren algoritem za določen problem. Po drugi strani je veliko algoritmov v bistvu kopija že prej obstoječih, vendar v drugi preobleki. Zaradi tega se priporoča, da uporabljamo algoritme, ki so najbolj razširjeni in obstajajo v literaturi že več kot desetletje in je možno preveriti njihovo ozadje. Zelo pomembno je omeniti NFL teorem [32], ki pravi, da ni univerzalnega iskalnega algoritma, ki bi uspešno deloval na vseh družinah problemov. Zato je potrebno vedno tudi raziskati kako so ostali raziskovalci rokovali z določenim problemov in kateri algoritmi so se najbolje obnesli na določenih problemih. Naslednji izziv je velikokrat nastavljanje nadzornih parametrov pri algoritmih. Skoraj vsak evolucijski algoritem ali algoritem inteligence rojev ima nekaj nadzornih parametrov, ki krmilijo njegovo delovanje tekom evolucijskega cikla. Včasih je potrebno vložiti veliko časa preden najdemo uspešno kombinacijo teh parametrov. Na srečo pa obstajajo tudi metode, kjer se parametri prilagajajo tekom evolucijskega cikla [33]. Dodatno je pri reševanju prostorsko zahtevnih problemov v splošnem težko ovrednotiti koliko dobra je trenutno najboljša najdena rešitev in ali sploh obstaja boljša. S tega stališča nam lahko izziv predstavlja tudi določitev zaustavitvenega pogoja, ki je navadno definiral z maksimalnim številom ovrednotenj kriterijske funkcije.

4 Primer avtomatskega načrtovanja delovnega časa

V nadaljevanju bomo predstavili praktični primer uporabe algoritmov računske inteligence za namen avtomatskega načrtovanja delovnih časov. V mnogih poklicih narava dela ki ga opravljajo, zahteva, da zaposlenih delajo v različnih izmenah, pri čemer je lahko tudi trajanje same izmene dinamično ali prilagojeno. Tipični poklici kjer je to izrazito vidno so zdravstvena oskrba, storitve varovanja, logistika in transport, proizvodnje in različni podporni centri. Poleg same narave dela določenega poklica je potrebno pri načrtovanju delovnega časa upoštevati tudi mnogo formalnih zahtev, da je lahko nek sestavljen urnik izvedljiv tudi v praksi ter pokriva vse poslovne kot tudi zakonske zahteve in omejitve. V splošnem delovni čas v posameznih organizacijah načrtujejo upravljalski oz. vodstveni kadri, ki dodobra poznajo zahteve iz vidika poslovanja kot tudi imajo bogate izkušnje pri načrtovanju

delovnega časa. Priprava načrta delovnega časa je v splošnem kompleksen problem, ki od načrtovalca zahteva visoko stopnjo napora. Nemalokrat se lahko tudi prepeti, da se nenadoma spremenijo poslovne zahteve ali pa se lahko zaradi različnih razlogov soočamo s pomanjkanjem človeških virov. V takih primerih morajo načrtovalci reagirati hitro in poskrbeti za čim učinkovitejšo naslovitev problema ter prilagoditi načrt delovnega časa, kar navadno sila stresno opravilo. Opisana problematika je več kot primeren kandidat za naslovitev z uporabo računalniških sistemov, saj nam razpoložljiva računska moč in napredni inteligentni algoritmi omogočajo reševanje takšnih problemov z manj potrebnega vloženega truda, s čimer lahko v različnih situacijah hitreje in učinkoviteje pridemo do ustrezne rešitve oz. načrta delovnega časa. V nadaljevanju bomo predstavili rešitev za splošno načrtovanje delovnega časa zaposlenih na delovna mesta glede na podane kriterije končnega uporabnika.

4.1 Opredelitev problema

Načrtovanje delovnega časa nemalokrat predstavlja veliko težavo saj je pri načrtovanju potrebno upoštevati specifike organizacije kot tudi razpoložljive človeške vire in nenazadnje zakonske omejitve. Načrtovanje in razvoj rešitve, ki omogoča avtomatsko načrtovanje delovnega časa predstavlja kompleksen problem, ki je lahko v primeru načrtovanja delovnega časa za večja podjetja težko rešiljiv ali celo nerešljiv. Glede na različne specifike podjetij in druge omejitve želimo razviti kar se da »splošno« rešitev, ki omogoča načrtovanje delovnega časa ob upoštevanju specifičnih želja in zahtev končnega uporabnika. Rešitev mora torej omogočati visoko stopnjo prilagodljivosti pri čemer mora upoštevati sledeče:

- Podjetje ima lahko več poslovnih enot, vsaka poslovna enota pa lahko ima več zaposlenih.
- Vsaka poslovna enota ima določeno za vsak dan koliko zaposlenih potrebuje za določeno izmeno in določeno znanje oz. delovno mesto.
- Vsak zaposleni ima eno ali več znanj oz. opravlja delo na različnih delovnih mestih (blagajnik, prodajnik, skladiščnik...). Vsak zaposleni ima naziv in seznam znanj pri čemer ima vsak posameznik eno ali več znanj, ki so primarne ter poljubno mnogo sekundarnih znanj s katerimi lahko opravlja delo na določenem delovnem mestu. Na primer zaposleni Jože Novak, je lahko blagajnik in prodajnik, izjemoma pa je lahko tudi skladiščnik. Rešitev ga torej mora primarno skušati dodeliti na delovni mesti blagajnika ali prodajnika, izjemoma pa ga lahko tudi razporedi na delovno mesto skladiščnika. Dodatno rešitev upošteva za vsakega zaposlenega tudi kdaj in koliko lahko dela oz. kdaj dela ne more opravljati (dopust, bolniška, maksimalno število ur na posamezno izmeno). Vsak zaposleni ima tudi stanje ur na začetku meseca, ter število ur, ki jih mora opraviti v mesecu, pri čemer ima lahko vsak delavec za opraviti različno število ur (dopusti, bolniške)
- Podjetje ima definiran šifrant izmen, na primer: dopoldan (8.00 -14.00), popoldan (14.00 – 22.00
- Podjetje ima definiran šifrant vrste dela (redno delo, nedeljsko delo, praznično delo...).

Tabela 1 prikazuje primer definicije zahtev glede števila zaposlenih na določenem delovnem mestu v posamezni poslovni enoti na določen dan.

Tabela 1: Primer vhodnih zahtev za avtomatsko načrtovanje delovnega časa

Poslovna enota	Datum	Vrsta dela	Izmena	Zaposleni
PE 1	1.10.2022	Redno delo	Dopoldan	2 blagajnika
		Redno delo	Dopoldan	3 prodajniki
		Redno delo	Popoldan	1 blagajnik
	2.10.2022	Nedeljsko delo	Dopoldan	2 blagajnika
PE 2	1.10.2022	Redno delo	Dopoldan	1 blagajnik

Glede na to, da želimo čim višjo prilagodljivost rešitve je smiselno tudi, da ta upošteva oz. omogoča delno zapolnitev delovnega urnika kar končnemu uporabniku omoča, da vnaprej določi kateri zaposleni bo na kateri dan

delal na nekem delovnem mestu in v izbrani poslovni enoti. Tako »zasedene« izmene razvita rešitev ne sme več spreminjati, seveda pa jih mora vključiti oz. upoštevati pri izračunu morebitnih prekoračitev omejitev.

Pogosto se dogodi, da imajo podjetja premalo na voljo premalo človeških virov glede na potrebe poslovanja. Posledično prihaja do nadur oz. do kršitev določenih omejitev. Rešitev mora tako omogočati, da lahko posamezno podjetje definira za določeno kršitev nekakšno kazen, cilj rešitve pa je seveda, da je seštevek morebitnih kazni čim manjši ob pogoju da je izdelan načrt dela še vedno izvedljiv tudi v praksi. Za lažjo predstavo si poglejmo primer. V primeru, da rešitev oblikuje načrt dela na način, da se nek zaposleni ponovno vrne na delo 10 ur po oddelani izmeni se v tem primeru krši minimalni dnevni počitek med dvema izmenama, ki znaša 12 ur. Če podjetje definira, da je 1 ura kršitve dnevnega počitka 50 točk kazni, pomeni da je rešitev pri načrtovanju prekršila dnevni počitek za 2 uri, kar znese 100 točk kazni. Pri tem je potrebno vse kršitve omejitev izračunavati na za vsakega zaposlenega posebej. Prav tako je pomembno, da so takšne morebitne kazni čimbolj enakomerno razporejene med zaposlene, saj s tem posledično prihaja do preobremenitev določenih zaposlenih in potencialno več bolniških odsotnosti ter dodatnega primankljaja zaposlenih. Neuravnotežene kršitve omejitev na ravni posameznega zaposlenega pa tudi lahko privedejo do splošnega nazadovoljstva zaposlenih. Rešitev mora končnemu uporabniku omogočati, da določi število kazenskih točk za posamezno kršitev glede na svoje želje oz. potrebe. Tabela 2 prikazuje seznam omejitev, ki jih razvita rešitev upošteva pri načrtovanju, pripadajoče vrednosti kršitev podanih omejitev pa uporabi za namen načrtovanja načrta dela, ki bo kršil omejitve v čim manjšem obsegu. Prijagajanje kazenskih točk za posamezno omejitev, končnemu uporabniku omogoča, da določi z njegovega vidika bolj pomembne (višje število kazenskih točk) in manj pomembne (nižje število kazenskih točk) omejitve.

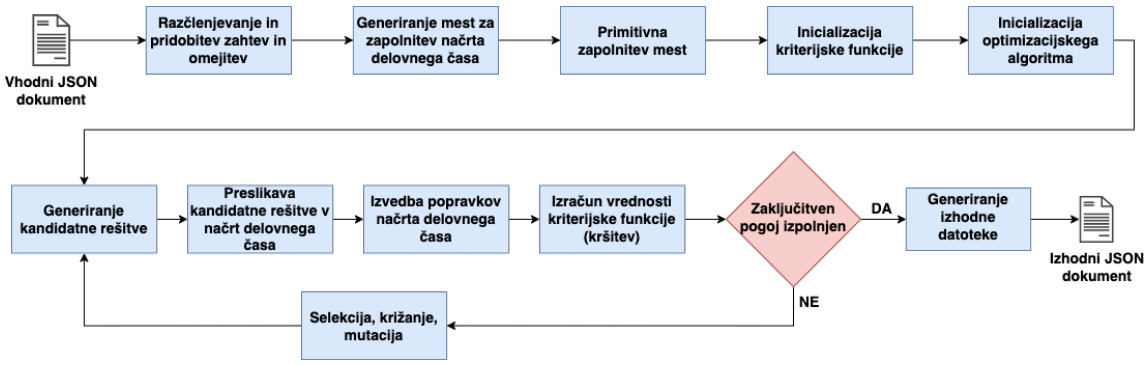
Tabela 2: Seznam kršitev podanih omejitev

Tip kršitve	Kazen
Kršitev za nepravilno vrsto dela	10 točk
Kršitev za prekoračitev delovne izmene	10 točk
Kršitev za nepravilno delovno mesto	5 točk
Kršitev za kršenje dnevnega počitka	20 točk
Kršitev za kršenje tedenskega počitka	20 točk
Kršitev za umestitev zaposlenega v drugo poslovno enoto	5 točk

Z definiranimi omejitvami lahko tako usmerjamo načrtovanje urnika v smeri zahtev podjetja, ne glede na kršitve pa gre v osnovi za mehke omejitve pri katerih bo končni rezultat rešitve izvedljiv načrt delovnega časa v kolikor seveda obstaja rešitev za dan problem. Poleg omenjenih omejitev mora razvita rešitev upoštevati tudi strožje omejitve katerih kršenje se odraža v obliki neizvedljivega načrta delovnega časa. V našem primeru smo si zadali dve takšni omejitvi in sicer razporeditev zaposlenega na dve delovni mesti v isti izmeni ter razporeditev zaposlenega na delovno mesto v terminu ko je ta odsoten (dopust, bolniška...).

4.2 Načrtovanje in implementacija rešitve

Skladno s predstavljenim problemom in zahtevami smo oblikovali rešitev temelječo na evolucijskem algoritmu oz. algoritmu inteligence rojev. Problem smo namreč zmodelirali na način, da nam omogoča uporabo praktično kateregakoli algoritma, ki operira z posamezniki definiranimi v obliki vektorja realnih števil. To nam posledično daje odlično izhodišče za eksperimentiranje z različnimi algoritmi brez, da bi pri tem morali posegati v samo zasnovo rešitve. Diagram (Slika 2) predstavlja konceptualno zasnovo naše rešitve za definiran problem avtomatskega načrtovanju delovnega časa.



Slika 2: Konceptualni diagram rešitve problema avtomatskega načrtovanja delovnega časa.
Vir: lasten.

Na vhodu prejemo JSON dokument z definiranimi zahtevami in omejitvami ter človeškimi viri, ki so nam na voljo na načrtovanje urnika dela. Podan dokument razčlenimo ter pridobimo vrednosti omejitev ter nabor zahtev. Sledi generiranje mest za zapolnitev načrta delovnega časa. V tem koraku vsako podano zahtevo preoblikujemo na način, da iz posamezne zahteve dobimo posamezna mesta za načrt delovnega časa. Vzemimo na primer zahtevo, da morata biti v ponedeljek 18.7.2022 v Poslovalnici 1, v jutranji izmeni, ki traja od 7.00 do 15.00 zaposlena dva blagajničarja. Takšno zahtevo preoblikujemo v dve mesti v načrtu delovnega časa kjer ima vsako mesto definirane enake zahteve kot so bile podane, le da lahko posamezno izmed tako oblikovanih mest možno zapolniti zgolj z enim zaposlenim. Po zaključku preoblikovanja zahtev in omejitev v obliko posameznih mest sledi zapolnitev morebitnih že v naprej določenih mest, ki so bila definirana že v vhodni datoteki. Takšnih mest rešitev ne sme premikati. Število mest, ki ostanejo nezapolnjena nam predstavlja dimenzijo našega problema – za toliko mest bo namreč naloga algoritma, da jih zapolni. Sledi inicializacija kriterijske funkcije. Ta skrbi za izračunavanje kazni v primeru morebitnih kršitev podanih omejitev načrtovanega urnika dela. V fazi inicializacije optimizacijskega algoritma, izbranemu algoritmu (v našem primeru smo preizkusili dva – diferencialno evolucijo in algoritem optimizacije z rojem delcev), nastavimo pripadajoče vrednosti parametrov. Število parametrov se od algoritma do algoritma razlikuje, kot tudi se razlikujejo njihove privzete vrednosti. Za primerno nastavitev vrednosti parametrov ni splošnega pravila kako jih nastaviti. Navadno je potrebno veliko eksperimentiranja, da dosežemo kar se da optimalne vrednosti parametrov, ki nam dajejo kar se da dobre rezultate. Poleg parametrov specifičnih za posamezen algoritem pa je potrebno nastaviti tudi velikost populacije ter maksimalno število ovrednotenj kriterijske funkcije. Na podlagi izkušenj je smiselno število populacije nastaviti na 15 v kolikor je dimenzija problema manjša od 30, na 30 v kolikor je dimenzija problema med 30 in 200 ter na 60 v kolikor je dimenzija problema večja od 200. Inicializiran algoritem zaženemo nam ciljnim problemomter pridobimo t.i. kandidatno rešitev v obliki vektorja realnih števil, katere na to preslikamo v obliko kjer posamezno mesto predstavlja delovno izmeno nekega zaposlenega. Sledi izvedba popravkov v tako pridobljenem načrtu delovnega časa kjer z zamenjavo posameznih zaposlenih na delovnih mesti dosežemo, da je ujemanje z omejitvami (delovno mesto oz. potrebna znanja)čim boljše. Nad tako »popravljenim« načrtom delovnega časa izračunamo morebitne kršitve vsakega posameznega zaposlenega za posamezno kršitev posebej. Vse kršitve istega tipa seštejemo ter nad njimi izračunamo uteženo vsoto kjer so kršitve definirane z večjim številom točk bolj obtežene kot tiste z nižjim številom točk. Sledi preverjanje zaključitvenega pogoja, ki je v našem primeru definiran z maksimalnim številom ovrednotenj kriterijske funkcije oz. z maksimalnim časom izvajanja. V kolikor pogoj ni izpolnjen se vrednost kriterjske funkcije pošlje algoritmu kjer sledi izvedba operacij selekcije, križanja in mutacije z namenom pridobitve nove kandidatne rešitve na podlagi ocene prejšnje kandidatne rešitve (vrednost kriterijske funkcije). V kolikor pa je zaključitven pogoj izpolnjen oblikuje rešitev načrta delovnega časa na podlagi do sedaj najboljše oblikovane rešitve, torej rešitve, ki je dosegla najnižje število kazenskih točk. Izhodna rešitev je zapisana v obliki JSON dokumenta.

Tako načrtovano rešitev smo implementirali z uporabo programskega jezika Python. Pri tem smo si pomagali z različnimi podpornimi knjižnicami: Click za implementacijo ukaznega vmesnika rešitve, luguru za zapisovanje dnevniških zapisov, jsonschema za validacijo vhodnih JSON dokumentov, prettytable za izpis rešitve v tabelarični obliki znotraj ukaznega vmesnika ter knjižnico NiaPy [33], ki vsebuje množico implementiranih algoritmov

delujočih po vzorih iz narave katere smo spridoma pouporabili za reševanje našega problema avtomatskega načrtovanja delovanega časa. Implementirano rešitev smo z uporabo knjižnjice PyInstaller zapakirali v izvedljivo datoteko (.exe), ki vključuje izvajalno Python okolje z vsemi potrebnimi knjižnicami, s čimer smo olajšali izvajanje razvite rešitve znotraj operacijskega sistema Windows.

4.3 Prikaz delovanja in ovrednotenje rešitve

Za namen interakcije z razvito rešitvijo smo razvili vmesnik z ukazno vrstico preko katerega lahko podamo vhodne parametre v skladu s katerimi je nato razvita rešitev pognana. Rešitev podpira tri zagonske parametre (zastavice) in sicer **-f** (ali **--file**) s katero lahko definiramo pot do vhodne JSON datoteke, **-t** (ali **--time**) s katero lahko definiramo maksimalen dovoljen čas izvajanja in **-o** (ali **--output**) s katero lahko definiramo pot kamor naj se shrani končna rešitev podanega problema avtomatskega načrtovanja delovnega časa.

Slika 3 prikazuje zagon razvite rešitve z uporabo vseh treh zagonskih parametrov.

```
.\kdajdelam-schedule.exe --file .\primeri\trgovina.json --time 1 --output rezultat.json
```

Slika 3: Primer zagona razvite rešitve.
Vir: lasten.

Z namenom objektivnega ovrednotenja in vpogledom v delovanje razvite rešitve smo opravili analizo delovanja na treh primerih realnih podjetij. Podjetja so si med seboj različna glede na panogo (veterinarstvo, trgovina, gostinski lokal) kot tudi glede na potrebe načrtovanja delovnega časa. V vseh izvedenih in analiziranih primerih so upoštevane omejitve pri načrtovanju urnika (glej Tabela 2).

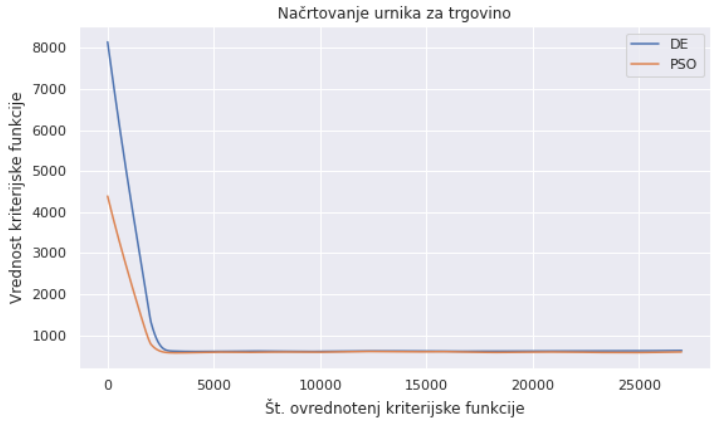
Za vsak analiziran primer smo izvedli dva zagona pri čemer smo v enem zagonu zadano nalogo reševali z uporabo algoritma diferencialne evolucije (angl. Differential Evolution, DE), v drugem pa z uporabo algoritma optimizacije z rojem delcev (angl. Particle Swarm Optimization, PSO).

4.3.1 Primer načrtovanja delovnega časa za trgovino

Primer načrtovanja delovnega časa za trgovino je najenostavnejši izmed vseh treh. Trgovina ima 8 zaposlenih in obratuje vse 6 dni v tednu od 8.00 – 6.00, torej 22 ur na dan. S tega vidika je ta primer še posebej specifičen. Načrt delovnega časa se je v tem primeru načrtoval za obdobje od 6.4.2020 do 11. 4.2020, torej za 1 delovni teden. Tabela 3 prikazuje najbolje oblikovan načrt delovnega časa ob uporabil algoritma DE in PSO. Iz tabele je razvidno, da je rešitev z uporabo algoritma DE oblikovala bolj optimalen načrt delovnega časa, saj je ta dosegel manj kazenskih točk kot tisti načrtovan z uporabo algoritma PSO. Poleg skupnega števila kazenskih točk, so iz tabele razvidne tudi dodeljene kazenske točke za posamezno kršenje omejitve. Kot lahko vidimo, sta oba algoritma bila enako kaznjena za kršitev omejitve za nepravilno vrsto dela (80 točk, kar se po tabeli kazenskih točk pretvori v 8 ur), pri ostalih pa pride do očitnega razhajanja v prid algoritmu DE.

Tabela 3: Prikaz kršitev najboljše načrtovanega delovnega časa za trgovino

Tip kazni	DE	PSO
Kazen za nepravilno vrsto dela	80 (8 ur)	80 (8 ur)
Kazen za prekoračitev delovne izmene	0	0
Kazen za nepravilno delovno mesto	0	50 (10 ur)
Kazen za kršenje dnevnega počitka	120 (6 ur)	320 (16 ur)
Kazen za kršenje tedenskega počitka	0	0
Kazen za umestitev zaposlenega v drugo poslovno enoto	0	0
Skupaj kazni	200	450



Slika 4: Potek zmanjševanja vrednosti kriterijske funkcije (kazni) pri načrtovanju urnika za trgovino.
Vir: lasten.

Ker je reševanje tako kompleksnih problemov lahko pogosto časovno zahtevna naloga, je pogosto bolj pomembno dobiti dovolj uporabno rešitev v čim krajšem času, kot pa dobiti malenkost boljšo rešitev vendar je za to potrebno veliko več časa. Slika 4 prikazuje potek optimizacije avtomatskega načrtovanja delovnega časa z algoritmom DE in PSO. Na *x* osi so števila ovrednotenj kriterijske funkcije na *y* osi pa vrednost te pri posameznem ovrednotenju. Iz vrednosti v začetni fazi optimizacije je razvidno, da so rešitve algoritma PSO bolj optimalne kot rešitve algoritma DE. Vidimo lahko, da oba algoritma največji del optimizacije načrta delovnega časa opravita v začetnih 2500 ovrednotenjih kriterijske funkcije, kar s stališča pridobitve čim boljše rešitve v čim krajšem času, nobenemu izmed algoritmov ne prinaša posebne prednosti. Na dolgi rok algoritem DE sicer uspe malenkostno izboljšati rešitev v primerjavi z algoritmov PSO, vendar ne občutno.

Slika 5 prikazuje primer izpisa tedenskega načrta dela ustvarjenega s pomočjo razvite rešitve ob uporabi optimizacijskega algoritma DE. Izpis je zgolj demonstracijske narave, končna rešitev pa je izvožena v obliki JSON dokumenta, ki omogoča nadaljno strojno obdelavo in poljuben izpis.

Monday (06.04)	Tuesday (07.04)	Wednesday (08.04)	Thursday (09.04)	Friday (10.04)	Saturday (11.04)
[08:00–16:00]:Zaposleni 1	[08:00–16:00]:Zaposleni 1	[08:00–16:00]:Zaposleni 4	[08:00–16:00]:Zaposleni 8	[08:00–16:00]:Zaposleni 4	[08:00–16:00]:Zaposleni 4
[08:00–16:00]:Zaposleni 2	[08:00–16:00]:Zaposleni 2	[08:00–16:00]:Zaposleni 5	[08:00–16:00]:Zaposleni 6	[08:00–16:00]:Zaposleni 5	[08:00–16:00]:Zaposleni 5
[08:00–16:00]:Zaposleni 3	[08:00–16:00]:Zaposleni 5	[08:00–16:00]:Zaposleni 2	[08:00–16:00]:Zaposleni 5	[08:00–16:00]:Zaposleni 2	[08:00–16:00]:Zaposleni 2
[14:00–20:00]:Zaposleni 4	[14:00–20:00]:Zaposleni 8	[14:00–20:00]:Zaposleni 8	[14:00–20:00]:Zaposleni 1	[14:00–20:00]:Zaposleni 8	[08:00–16:00]:Zaposleni 6
[14:00–20:00]:Zaposleni 5	[14:00–20:00]:Zaposleni 3	[14:00–20:00]:Zaposleni 1	[14:00–20:00]:Zaposleni 2	[14:00–20:00]:Zaposleni 6	[14:00–20:00]:Zaposleni 1
[20:00–06:00]:Zaposleni 6	[20:00–06:00]:Zaposleni 6	[14:00–20:00]:Zaposleni 3	[14:00–20:00]:Zaposleni 3	[14:00–20:00]:Zaposleni 1	[14:00–20:00]:Zaposleni 8
[20:00–06:00]:Zaposleni 7	[20:00–06:00]:Zaposleni 7	[14:00–20:00]:Zaposleni 6	[14:00–20:00]:Zaposleni 7		[14:00–20:00]:Zaposleni 3
	[20:00–06:00]:Zaposleni 6				[14:00–20:00]:Zaposleni 7

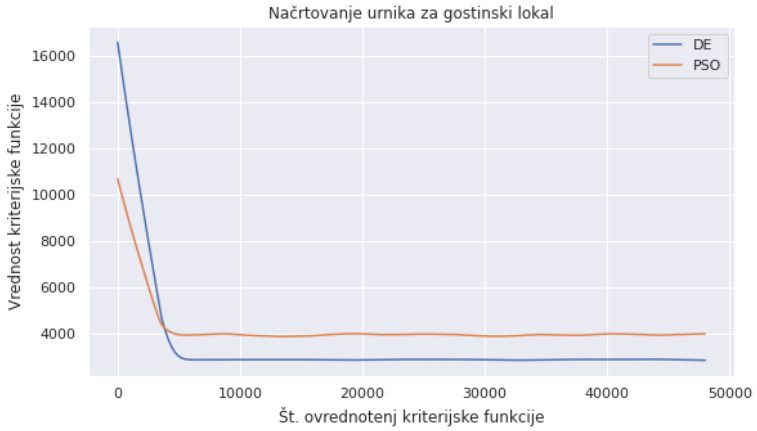
Slika 5: Primer avtomatsko načrtovanega delovnega časa za en teden ob uporabi algoritma DE.
Vir: lasten.

4.3.2 Primer načrtovanja delovnega časa za gostinski lokal

Gostinski lokal ima skupno 6 zaposlenih in obratuje vse dni v tednu od 7.00 do 22.00 ure. Načrt delovnega časa se je načrtoval za obdobje med 30.12.2019 in 2.2.2020 (5 polnih tednov) kar lahko pretvorimo v 150 mest v katere je bilo potrebno dodeliti 6 zaposlenih. Specifika pri načrtovanju urnika za ta primer je prekrivanje posameznih izmen dela ter različne dolžine posameznih izmen. Tabela 4 prikazuje najboljši načrt delovnega časa ob uporabi algoritma diferencialne evolucije in algoritma optimizacije z rojem delcev. Iz tabele je razvidno, da je algoritem DE pri enakih pogojih uspel načrtovati optimalnejši delovni čas, saj je seštevek kazni manjši kot pri rešitvi pridobljeni z uporabo algoritma PSO.

Tabela 4: Prikaz kršitve najboljše načrtovanega delovnega časa za gostinski lokal

Tip kazni	DE	PSO
Kazen za nepravilen tipa dela	1405 (140,5 ur)	1400 (140 ur)
Kazen za prekoračitev delovne izmene	0	0
Kazen za nepravilno delovno mesto	0	0
Kazen za kršenje dnevnega počitka	210 (10,5 ur)	380 (19 ur)
Kazen za kršenje tedenskega počitka	0	0
Kazen za umestitev zaposlenega v drugo poslovno enoto	0	0
Skupaj kazni	1615	1780



Slika 6: Potek zmanjševanja vrednosti kriterijske funkcije (kazni) pri načrtovanju urnika za gostinski lokal. Vir: lasten.

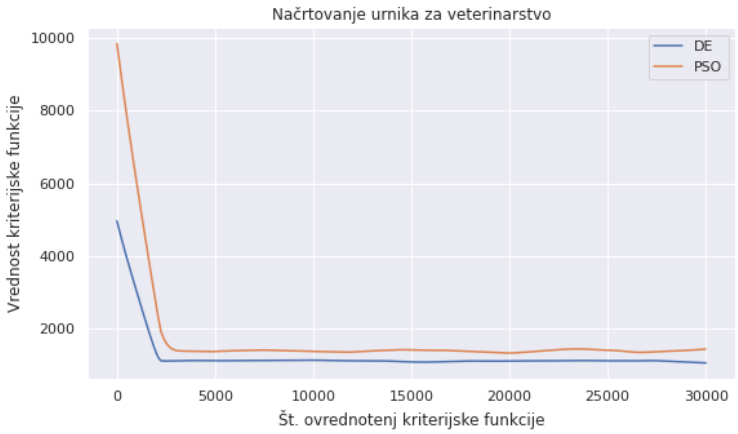
Iz Slika 6 lahko razberemo da sicer algoritem PSO v začetni fazi delovanje načrtuje bolj optimalne rešitve kot algoritem DE, vendar oba glavnino optimizacije opravita do približno 5000. ovrednotenja kriterijske funkcije. Posledično sta, gledano iz vidika pridobitve čim boljše rešitve v čim krajšem času, praktično enakovredno. Iz krivulij je razvidno, da v nadaljevanju DE algoritem načrtuje bolj optimalne rešitve kot algoritem PSO.

Primer načrtovanja delovnega časa za **za veterinarstvo**Tretji primer je primer načrtovanja delovnega časa za veterinarstvo. Veterinarstvo ima 5 zaposlenih in posluje od ponedeljka do petka med 8.00 do 20.00. Načrt delovnega časa se je oblikoval za obdobje med 1.4.2020 in 30.4.2020 (22 delovnih dni) kar lahko pretvorimo v 110 mest v katere je bilo potrebno dodeliti 5 zaposlenih. Iz Tabela 5 je razvidno, da je optimalnejši načrt dela sestavil optimizacijski algoritem DE. Sestavljen urnik je bil kaznjen zgolj za eno kategorijo kršitev in sicer je kršil omejitev za prekoračitev delovne izmene v obsegu 600 točk kar se pretvori v 60 ur. Rešitev pridobljena z algoritmo PSO je sicer v tej kategoriji kršitev dosegla manjšo kazen vendar je poleg te pridelala še znatno vsoto kazenski točk zaradi kršenja omejitve dnevnega počitka v obsegu 320 točk oz. 16 ur.

Tabela 5: Prikaz kršitve najboljše načrtovanega delovnega časa za veterinarstvo

Tip kazni	DE	PSO
Kazen za nepravilen tipa dela	0	0
Kazen za prekoračitev delovne izmene	600 (60 ur)	540 (54 ur)
Kazen za nepravilno delovno mesto	0	0
Kazen za kršenje dnevnega počitka	0	320 (16 ur)
Kazen za kršenje tedenskega počitka	0	0
Kazen za umestitev zaposlenega v drugo poslovno enoto	0	0
Skupaj kazni	600	860

Za razliko od poteka zmanjševanja vrednosti kriterijske funkcije v prejšnih dveh primerih je v tem primeru iz grafa razvidno, da so rešitve pridobljene z algoritmod DE skozi celoten potek optimizacije vidno boljše kot tiste pridobljene z uporabo algoritma PSO. Podobno kot pri prejšnjih dveh primerih, je tudi v tem točka kjer je opravljen večji del optimizacije načrta delovnega časa podobna pri obeh algoritmih.



Slika 7: Potek zmanjševanja vrednosti kriterijske funkcije (kazni) pri načrtovanju urnika za veterinarstvo. Vir: lasten.

5 Zaključek

Reševanje industrijskih problemov z uporabo računske inteligence se je in se še vedno izkazuje kot zelo uspešno pri reševanju raznovrstnih problemov. Ne glede na napredek pristopov, metod in tehnik računske inteligence pa se še vedno v veliki meri, pri vpeljavi teh soočamo z številnimi izzivi. Zahtevnost vpeljave nam sicer lajšajo programske knjižnjice in paketi, ki imajo v naprej definiran nabor raznovrstnih algoritmov kar nam omogoča lažjo eksperimentiranje pri snovanju rešitve za naš cilnji problem. V prispevku smo predstavili najpogostejše skupine algoritmov, ki se tipično uporabljajo za naslavljanje industrijskih problemov. Podali smo smernice za izbiro primernih algoritmov ter se dotaknili izzivov s katerimi se pogosto soočamo. Vpeljavo računske inteligence smo prikazali na primeru avtomatksega načrtovanja delovnega časa. Razvita rešitev se izkaže s svojo prilagodljivostjo glede na zahteve, želje in potrebe končnega uporabnika, rešitve načrtov dela, ki jih oblikuje pa so smiselne in izvedljive. Analizirane primere smo izvedli z uporabo dveh različnih optimizacijskih algoritmov, pri čemer se je v vseh primerih algoritem diferencialne evolucije izkazal za uspešnejšega kar še enkrat več potrjuje njegovo vsesplošno uporabnost.

Literatura

[1] Fister, I. (2017). Algoritmi računske inteligence za razvoj umetnega športnega trenerja. Univerza v Mariboru (Slovenia).
[2] Blum, C., & Merkle, D. (Eds.). (2008). Swarm intelligence: introduction and applications. Springer Science & Business Media.
[3] Terano, Toshiro, Kiyoji Asai, and Michio Sugeno. Fuzzy systems theory and its applications. Academic Press Professional, Inc., 1992.
[4] Engelbrecht, A. P. (2007). Computational intelligence: an introduction. John Wiley & Sons.
[5] Eiben, A. E., & Smith, J. E. (2003). Introduction to evolutionary computing (Vol. 53, p. 18). Berlin: springer.
[6] Dasgupta, Dipankar. "Advances in artificial immune systems." IEEE computational intelligence magazine 1.4 (2006): 40-49.
[7] FISTER, Iztok, 2019, Avtomatsko načrtovanje in vrednotenje klasifikacijskih cevovodov v bioinformatiki [na spletu]. Univerza v Mariboru, Fakulteta za zdravstvene vede.

[8] Fister Jr, I., Yang, X. S., Fister, I., Brest, J., & Fister, D. (2013). A brief review of nature-inspired algorithms for optimization. arXiv preprint arXiv:1307.4186.

[10] Hornby, Gregory, et al. "Automated antenna design with evolutionary algorithms." Space 2006. 2006. 7242.

[11]Ghoddusi, Hamed, Germán G. Creamer, and Nima Rafizadeh. "Machine learning in energy economics and finance: A review." Energy Economics 81 (2019): 709-727.

[12] Tayarani-N, Mohammad-H., Xin Yao, and Hongming Xu. "Meta-heuristic algorithms in car engine design: A literature survey." IEEE Transactions on Evolutionary Computation 19.5 (2014): 609-629.

[13] Nikolos, Ioannis K., Eleftherios S. Zografos, and Athina N. Brintaki. "UAV path planning using evolutionary algorithms." Innovations in intelligent machines-1. Springer, Berlin, Heidelberg, 2007. 77-111.

[14] Gao, Shangce, et al. "A state-of-the-art differential evolution algorithm for parameter estimation of solar photovoltaic models." Energy Conversion and Management 230 (2021): 113784.

[15] Saavedra-Moreno, B., et al. "Seeding evolutionary algorithms with heuristics for optimal wind turbines positioning in wind farms." Renewable Energy 36.11 (2011): 2838-2844.

[16] Jursa, René, and Kurt Rohrig. "Short-term wind power forecasting using evolutionary algorithms for the automated specification of artificial intelligence models." International Journal of Forecasting 24.4 (2008): 694-709.

[17] Podgorelec, Vili, et al. "Digital Transformation Using Artificial Intelligence and Machine Learning: An Electrical Energy Consumption Case." International Conference "New Technologies, Development and Applications". Springer, Cham, 2022.

[18] Fister, Iztok, Iztok Fister Jr, and Dušan Fister. Computational intelligence in sports. Cham: Springer, 2019.

[19] Pena-Reyes, Carlos Andrés, and Moshe Sipper. "Evolutionary computation in medicine: an overview." Artificial Intelligence in Medicine 19.1 (2000): 1-23.

[20] Slowik, Adam, and Halina Kwasnicka. "Evolutionary algorithms and their applications to engineering problems." Neural Computing and Applications 32.16 (2020): 12363-12379.

[21] Odili, Julius, et al. "A comparative evaluation of swarm intelligence techniques for solving combinatorial optimization problems." International Journal of Advanced Robotic Systems 14.3 (2017): 1729881417705969.

[22] Zhang, Shuzhu, et al. "Swarm intelligence applied in green logistics: A literature review." Engineering Applications of Artificial Intelligence 37 (2015): 154-169.

[30] Gondro, Cedric, and B. P. Kinghorn. "Application of evolutionary algorithms to solve complex problems in quantitative genetics and bioinformatics." Guelph: University of Guelph (2008).

[31] Tzanetos, Alexandros, Iztok Fister Jr, and Georgios Dounias. "A comprehensive database of Nature-Inspired Algorithms." Data in brief 31 (2020): 105792.

[32] Wolpert, David H., and William G. Macready. "No free lunch theorems for optimization." IEEE transactions on evolutionary computation 1.1 (1997): 67-82.

[33] Vrbančič G, Brezočnik L, Mlakar U, Fister D, Fister I. NiaPy: Python microframework for building nature-inspired algorithms. Journal of Open Source Software. 2018 Mar 22;3(23):613.