UNAPREĐENJE RASPOREĐIVANJA POSLOVA I BALANSIRANJA OPTEREĆENJA U KLAUD OKRUŽENJU PRIMENOM METAHEURISTIKA INTELIGENCIJE ROJEVA

By Ivana Štrumberger

WORD COUNT 48135 TIME SUBMITTED 09-DEC-2019 10:33AM
PAPER ID 52812544

UNIVERZITET SINGIDUNUM
BEOGRAD
DEPARTMAN ZA POSLEDIPLOMSKE
STUDIJE

DOKTORSKA DISERTACIJA

UNAPREĐENJE RASPOREĐIVANJA POSLOVA I BALANSIRANJA OPTEREĆENJA U KLAUD OKRUŽENJU PRIMENOM METAHEURISTIKA INTELIGENCIJE ROJEVA

Mentor: prof. dr Milan Tuba

Student: Ivana Štrumberger
Broj indeksa: 460048/2018

Beograd, 2019. godina



SINGIDUNUM UNIVERSITY
BELGRADE
DEPARTMENT FOR POSTGRADUATE STUDIES

PhD Thesis

IMPROVEMENTS OF TASK SCHEDULING AND LOAD BALANCING IN CLOUD ENVIRONMENT BY SWARM INTELLIGENCE METAHEURISTICS

Mentor: Professor Milan Tuba

Student: Ivana Strumberger
Index number: 460048/2018

Belgrade, 2019.

MENTOR:

dr Milan Tuba, redovni profesor
UNIVERZITET SINGIDUNUM

ČLANOVI KOMISIJE:

dr Đorđe Obradović, vanredni profesor
UNIVERZITET SINGIDUNUM

dr Milan Rapaić, vanredni profesor
UNIVERZITET U NOVOM SADU, FAKULTET TEHNIČKIH NAUKA

Posvećeno porodici i prijateljima

Unapređenje raspoređivanja poslova i balansiranja opterećenja u klaud okruženju primenom metaheuristika inteligencije rojeva

Sažetak – Klaud računarstvo pripada grupi novijih računarskih paradigmi, koja se poput paradigme mrežnog računarstva, bazira na grupisanju resursa i na korišćenju mrežnih i Internet tehnologija. U opštem smislu, klaud računarstvo se odnosi na novi način isporuke računarskih resursa u vidu usluge, gde se pod resursima podrazumeva gotovo sve, od podataka i softvera, do hardverskih komponenti, kao što su procesirajući elementi, memorija i skladišta.

Klaud računarstvo je aktuelna i važna multidisciplinarna oblast, o čemu svedoči veliki broj objavljenih radova u vrhunskim međunarodnim časopisima i prikazanih na najznačajnijim svetskim skupovima. Na osnovu naučnih rezultata prikupljenih u objavljenim radovima iz ovog domena, može da se zaključi da u klaud okruženju postoji veliki broj izazova i problema, za čije rešavanje mogu da se pronađu bolje metode, tehnike i algoritmi. Jedan od najvažnijih izazova savremenog klaud okruženja je raspoređivanje zahteva krajnjih korisnika za izvršavanje na ograničenom skupu raspoloživih resursa (virtuelnih mašina). Problem raspoređivanja na klaudu odnosi se na definisanje rasporeda izvršavanja zadataka na ograničenom skupu raspoloživih resursa uzimajući pritom u obzir potencijalna ograničenja i funkciju cilja koju je potrebno optimizovati.

Raspoređivanje poslova vrše algoritmi raspoređivanja, koji mogu da se podele na statičke i dinamičke. U slučaju statičkog raspoređivanja, gde se poslovi ne mogu dinamički prebacivati sa preopterećnih na manje opterećene virtuelne mašine, zadaci se raspoređuju za izvršavanje na raspoložive virtuelne mašine pre početka izvršavanja. S druge strane, primenom metoda dinamičkog raspoređivanja, koje je u literaturi poznato pod nazivom balansiranje opterećenja, vrši se preraspodela poslova između aktivnih virtuelnih mašina tokom samog izvršavanja programa raspoređivanja. Preraspodela se vrši tako što se zadaci sa virtuelnih mašina koje imaju veće opterećenje dinamički prebacuju za izvršavanje na virtuelnim mašinama koje imaju manje opterećenje. Za potrebe dinamičkog raspoređivanja koriste se uglavnom heurističke i metaheurističke optimizacione metode i algoritmi, koji postižu dobre rezultate.

Problemi raspoređivanja poslova i balansiranja opterećenja na klaudu pripadaju grupi NP teških kombinatornih i/ili globalnih problema sa ili bez ograničenja. Na osnovu publikovanih rezultata u relevantnim literaturnim izvorima, vidi se da su metaheuristike inteligencije rojeva, koje spadaju u grupu prirodom-inspirisanih algoritama, uspešno testirane na benčmark problemima i primenjivane na praktičnim NP teškim optimizacionim problemima (globalnim i kombinatornim) i da mogu da postignu bolje rezultate u smislu brzine konvergencije i kvaliteta rešenja, od drugih metoda, tehnika i algoritama. Polazeći od navedenog, u ovom radu je ispitivano da li je moguće dalje unaprediti rešavanja problema raspoređivanja poslova i balansiranja opterećenja na klaudu primenom metaheuristika inteligencije rojeva.

Tokom sprovedenog istraživanja, unapređeno je i adaptirano više metaheuristika inteligencije rojeva za rešavanje problema raspoređivanja poslova i balansiranja opterećenja u klaud okruženju. U disertaciji su detaljno prikazane implementacije dva unapređena algoritma rojeva - algoritma optimizacije monarh leptirovima i algoritma optimizacije jatom kitova. Za potrebe testiranja, rešavana su dva modela raspoređivanja poslova na klaudu. Prvi model, koji pripada grupi jednokriterijumske optimizacije, uzima u obzir minimizaciju vremena izvršavanja svih zadataka na klaudu, dok drugi, višekriterijumski model uzima u obzir minimizaciju vremena izvršavanja svih zadataka na klaudu i budžeta, tj. troškova za izvršavanje svih zahteva krajnjih korisnika. Simulacije su vršene u robusnom okruženju CloudSim simulatora i oba algoritma su testirana sa skupom veštačkih podataka, generisanih u okviru CloudSim platforme, i realnih podataka, koji su preuzeti iz globalno dostupne benčmark baze.

Osim testiranja za praktičan izazov na klaudu, da bi se preciznije utvrdila unapređenja modifikovanih metaheuristika u odnosu na osnovne verzije, obe metaheuristike su verifikovane i testiranjima na standardnim skupovima benčmark funkcija za globalnu optimizaciju bez ograničenja. Upoređivanjem generisanih rezultata (kvalitet rešenja i brzina konvergencije) sa rezultatima najboljih poznatih metaheuristika i heuristika iz literature, koje su primenjivane na iste instance problema (na praktičan problem raspoređivanja na klaudu i benčmark testove), dokazan je kvalitet implementiranih algoritama, čime je potvrđena i osnovna hipoteza ovog rada da se rešavanje izazova raspoređivanja poslova i balansiranja opterećenja u klaud okruženju mogu

dalje unaprediti primenom metaheuristika inteligencije rojeva.

Ključne reči: optimizacija, metaheuristike inteligencije rojeva, klaud računarstvo, raspoređivanje poslova, balansiranje opterećenja, NP teški problemi.

Naučna oblast: Računarstvo

Uža naučna oblast: Veštačka inteligencija

UDK broj: 004 (004.8) **CERIF:** P170, P176

Improvements of Task Scheduling and Load Balancing in Cloud Environment by Swarm Intelligence Metaheuristics

Abstract – Cloud computing paradigm, like the paradigm of network computing, is based on the clustering of resources and on the usage of network and Internet technologies. In general, the cloud computing refers to a new way of delivering computing resources in a form of service, such as data, software and hardware components (processing elements, memory and storage).

Cloud computing is a current and important multidisciplinary field, followed by a large number of published papers in the state-of-the-art international journals, as well as in the proceedings of the world-renowned international conferences. Based on the scientific results which have been gathered from the published papers in this domain, it can be concluded that there are many challenges and problems in the cloud environment, which can be more efficiently tackled by applying improved methods, techniques and algorithms. One of the most important challenges in cloud computing is scheduling of end users' requests on a limited set of available resources (virtual machines). A scheduling problem in cloud environment can be defined as an execution schedule of tasks on a limited set of available resources, taking into account the potential constraints and objective function.

Task scheduling is performed by scheduling algorithms, which can be divided into static and dynamic. In the case of a static scheduling, where it is not possible to dynamically switch tasks from over utilized to underutilized virtual machines, tasks are being allocated for execution on available virtual machines before the scheduling algorithm execution. In the case of the dynamic scheduling methods, which are known in the literature as load balancing approaches, a workload allocation between active virtual machines is being performed during the scheduling algorithm run-time. Requests' redistribution is executed by dynamically switching from over utilized to less utilized virtual machines. Heuristics and metaheuristics optimization methods are mostly used for dynamic scheduling, where they have achieved great results.

Task scheduling and load balancing problems in cloud computing belong to the group of NP hard combinatorial and/or global problems with or without constraints.

Based on the published results in the relevant literature, it can be concluded that the swarm intelligence metaheuristics have been successfully tested on benchmark and practical NP hard optimization problems, and that they have achieved better results in terms of convergence speed and the solutions' quality, than other methods, techniques and algorithms. In experiments performed for the purpose of this doctoral thesis, it is examined whether it is possible to further improve the task scheduling and load balancing in cloud computing environment by applying swarm intelligence metaheuristics.

During the experimental research, several swarm intelligence metaheuristics were improved and adapted for solving task scheduling and load balancing problems in cloud environment. The implementation of two improved swarm intelligence algorithms - the monarch butterfly optimization algorithm and whale optimization algorithm - are presented in the dissertation. Two task scheduling models have been considered. The first model belongs to the group of single-objective optimization, where the criteria is the minimization of execution time of all tasks on the cloud. The second model belongs to the category of multi-objective optimization, that considers minimizing the execution time of all tasks on the cloud and the total cost of utilized cloud resources. The simulations were performed in a robust CloudSim simulator environment, and both algorithms were tested against the set of artificial data (generated within the CloudSim platform), and real data (retrieved from a globally available benchmark database).

Besides testing swarm intelligence algorithms on practical cloud problems, in order to more accurately measure the improvements of modified metaheuristics over the original versions, both improved algorithms have also been validated by conducting simulations on standard benchmark sets for unconstrained global optimization. By comparing generated results (solution quality and convergence rate) of improved metaheuristics with the results of the best known metaheuristics and heuristics from the literature, that have been applied to the same problem instances (for practical cloud scheduling problem and benchmark tests), the quality of the proposed metaheuristics was validated, thus confirming the basic hypothesis of this dissertation that the tasks scheduling and load balancing in cloud computing can be further enhanced by utilizing swarm intelligence algorithms.

Keywords: optimization, swarm intelligence metaheuristics, cloud computing, task scheduling, load balancing, NP hard problems.

Scientific field: Computer Science

Scientific subfield: Artificial Intelligence

UDC number: 004 (004.8) **CERIF:** P170, P176

Sadržaj

Spisak slika **xiii**

Spisak tabela **xv**

Spisak algoritama **xvii**

1 Uvod **1**

1.1 Predmet i cilj istraživanja, hipoteze i naučni doprinos 4

1.2 Struktura doktorske disertacije 7

2 Pojam i koncept klaud računarstva **10**

2.1 Motivacija i potreba za klaud računarstvom 12

2.2 Tehnologija virtuelizacije i hiper-konvergirana infrastruktura 14

2.3 Definisanje klaud računarstva i osnovni modeli 17

2.4 Arhitektura klaud računarstva 22

3 Pojam i kategorizacija optimizacionih problema **25**

3.1 Taksonomija optimizacionih problema 26

3.1.1 Kategorizacija prema tipu varijabli 27

3.1.2 Podela na osnovu ograničenja 29

3.1.3 Kategorizacija prema broju funkcija cilja 31

3.2 Algoritmi i klase kompleksnosti 32

4 Modeli raspoređivanja poslova i balansiranja opterećenja na klaudu **35**

4.1 Raspoređivanje poslova u klaud okruženju i algoritmi 35

4.2 Model raspoređivanja resursa u klaud okruženju sa jednom funkcijom cilja 39

4.3 Model raspoređivanja resursa u klaud okruženju sa više funkcija cilja 42

5 Optimizacioni algoritmi **45**

5.1 Kategorizacija optimizacionih algoritama 46

5.2 Heuristike 49

5.3 Metaheurisitke 51

5.3.1 Metaheuristike koje nisu inspirisane prirodom 55

5.3.2 Metaheuristike koje su inspirisane prirodom 58

6 Metaheuristike inteligencije rojeva 62

6.1 Pojam i osnovni principi metaheuristika rojeva 62

6.2 Prikaz osnovnih implementacija algoritama rojeva koji su unapređeni 66

6.2.1 Algoritam optimizacije monarh leptirovima 66

6.2.2 Algoritam optimizacije jatom kitova 70

6.3 Pregled literature 75

7 Primena hibridnog MBO algoritma na rešavanje problema raspoređivanja poslova i balansiranja opterećenja u klaud računarstvu 80

7.1 Nedostaci osnovnog MBO algoritma 81

7.2 Prikaz hibridizovanog MBO algoritma 82

7.3 Testiranje MBO-ABC hibridnog pristupa na standardnim benčmark problemima 86

7.3.1 Benčmark funkcije i podešavanje parametara 86

7.3.2 Rezultati simulacija i diskusija 88

7.4 Testiranje MBO-ABC metaheuristike na problemu raspoređivanja poslova na klaudu 95

7.4.1 Prilagođavanje hibridne MBO metaheuristike za model raspoređivanja poslova 96

7.4.2 Prikaz CloudSim platforme 97

7.4.3 Simulacije sa veštačkim skupom podataka 100

7.4.4 Simulacije sa realnim skupom podataka 105

7.5 MBO-ABC zaključak 111

8 Implementacija hibridnog WOA algoritma za rešavanje problema raspoređivanja poslova i balansiranja opterećenja u klaud računarstvu 113

8.1 Nedostaci osnovne WOA metaheuristike 114

8.2 Prikaz hibridizovanog WOA algoritma 115

8.2.1 ABC mehanizam eksploracije i dodatni kontrolni parametri . . 117

8.2.2 FA jednačina pretrage 118

8.2.3 Inicijalizacija populacije i WOA-AEFS pseudo kod 119

8.3 Simulacije na standardnim benčmark problemima 122
8.3.1 Matematičke formulacije standardnih benčmark funkcija . . . 122
8.3.2 Podešavanje kontrolnh parametara hibridnog WOA-AEFS algoritma 125
8.3.3 Komparativna analiza i diskusija 126
8.4 Simulacije WOA-AEFS metaheuristike za problem raspoređivanja poslova na klaudu 132
8.4.1 Simulacije sa pravim skupom podataka 132
8.4.2 Simulacije sa veštačkim skupom podataka 139
8.5 WOA-AEFS zaključak 147

9 Zaključak **149**

10 Literatura **153**

Spisak slika

1 Predviđanje vrednosti globalnog javnog klaud tržišta prema kompaniji Gartner; BpaaS—business process as a service; CMaSS—cloud management and society services. 12

2 Procenat EU kompanija koji je koristio neku od klaud usluga tokom 2014. i 2018. godine. 13

3 Slojevita struktura tip 1 hipervizora 16

4 Slojevita struktura tip 2 hipervizora 16

5 Tri osnovne grupe klaud računarstva prema tipu usluga 19

6 Stepen odgovornosti i kontrole nad klaud resursima 20

7 Tipovi klauda na osnovu kriterijuma modela isporuke 22

8 Model sistema raspoređivanja zadataka u klaud okruženju 36

9 Brzina konvergencije MBO-ABC i MBO za F2 benčmark sa $D = 30$ i $D = 60$. 92

10 Brzina konvergencije MBO-ABC i MBO za F6 benčmark sa $D = 30$ i $D = 60$. 92

11 Primer šeme kodiranja rešenja 96

12 CloudSim arhitektura 98

13 Komparativna analiza makespan indikatora — MBO-ABC vs. MBO za stopu pristizanja od 10 poslova u sekundi sa veštačkim skupom podataka 103

14 Komparativna analiza makespan indikatora —MBO-ABC vs. ostali pristupiza stopu pristizanja od 10 poslova u sekundi sa veštačkim skupom podataka 104

15 Komparativna analiza—MBO-ABC vs. MBO sa HPC2N bazom realnih podataka sa manjem brojem poslova 107

16 Komparativna analiza—MBO-ABC vs. MBO sa HPC2N bazom realnih podataka sa manjem brojem poslova 107

17 MBO-ABC vs. MBO brzina konvergencije za instancu problema sa 2000 poslova sa HPC2N realnim skupom podataka 108

18 Komparativna analiza—MBO-ABC vs. MSDE vs. RR u simulacijama sa HPC2N realnim skupom podataka za manji broj poslova 109

19 Komparativna analiza—MBO-ABC vs. MSDE vs. RR u simulacijama sa HPC2N realnim skupom podataka za veći broj poslova 110
20 Graf brzine konvergencije WOA-AEFS vs. WOA: F8 benčmark 129
21 Graf brzine konvergencije WOA-AEFS vs. WOA: F21 benčmark . . . 129
22 Komparativna analiza—WOA-AEFS vs. WOA korišćenjem NASA iPSC realnih podataka za mali i veliki broj poslova 135
23 Grafikon brzine konvergencije WOA-AEFS vs. WOA korišćenjem NASA iPSC realnih podataka za 200 zahteva 135
24 Komparativna analiza—WOA-AEFS vs. drugi algoritmi korišćenjem NASA iPSC realnih podataka sa manjim brojem poslova 136
25 Komparativna analiza—WOA-AEFS vs. drugi algoritmi korišćenjem NASA iPSC realnih skupova sa većim brojem poslova 137
26 Komparativna analiza makespan funkcije cilja — WOA-AEFS vs. druge tehnike korišćenjem veštačkog skupa podataka (stopa dolaska 10) 142
27 Komparativna analiza makespan funkcije cilja — WOA-AEFS vs. druge algoritme korišćenjem veštačkog skupa podataka (stopa pristizanja 40) 142
28 Komparativna analiza troškova sa 200 zahteva i promenljivim rokom izvršavanja zadataka —WOA-AEFS vs. drugi algoritmi korišćenjem veštačkog skupa podataka 144
29 Komparativna analiza troškova sa 400 zahteva promenljivim rokom izvršavanja zadataka—WOA-AEFS vs. drugi algoritmi korišćenjem veštačkog skupa podataka 144
30 Komparativna analiza troškova sa 500 zahteva i promenljivim rokom izvršavanja zadataka —WOA-AEFS vs. drugi algoritmi korišćenjem veštačkog skupa podataka 145
31 Komparativna analiza narušavanja roka sa 200 zahteva —WOA-AEFS vs. drugi algoritmi korišćenjem veštačkog skupa podataka 146
32 Komparativna analiza narušavanja roka sa 400 zahteva —WOA-AEFS vs. drugi algoritmi korišćenjem veštačkog skupa podataka 146
33 Komparativna analiza narušavanja roka sa 500 zahteva —WOA-AEFS vs. drugi algoritmi korišćenjem veštačkog skupa podataka 147

Spisak tabela

1 Karakteristike benčmark funkcija korišćenih u simulacijama sa MBO-ABC algoritmom 86

2 Podešavanja parametara MBO i MBO-ABC algoritma 87

3 Komparativna analiza—Hibridizovani MBO-ABC vs. originalni MBO na benčmark funkcijama sa 30 dimenzija ($D = 30$) 89

4 Komparativna analiza—hibridni MBO-ABC vs. originalni MBO na benčmark funkcijama sa 60 dimenzija ($D = 60$) 91

5 Eksperimenti sa različitim vrednostima parametra *exh* MBO-ABC algoritma za benčmark funkcije sa 30 dimenzija 94

6 Eksperimenti sa različitim vrednostima parametra *dmt* MBO-ABC algoritma za benčmark funkcije sa 30 dimenzija 95

7 Parametri slučajno generisanog CloudSim okruženja u simulacijama sa veštačkim skupom podataka (MBO-ABC algoritam) 102

8 Unapređenje makespan vrednosti u odnosu na MBO u simulacijama sa veštačkim skupom podataka 103

9 Unapređenje makespan vrednosti u odnosu na druge metaheuristike u simulacijama sa veštačkim skupom podataka 105

10 CloudSim parametri za simulacije sa realnim skupom podataka (MBO-ABC algoritam) 106

11 Poboljšanja MBO-ABC u odnosu na originalan MBO u simulacijama sa HPC2N realnim skupom podataka 108

12 Komparativna analiza MBO-ABC sa MSDE i RR u simulacijama sa HPC2N realnim skupom podataka 110

13 Komparativna analiza DI indikatora u simulacijama sa HPC2N realnim skupom podataka – MBO-ABC vs. MBO, MSDE, RR 111

14 Osnovne Karakteristike unimodalnih benčmark funkcija 123

15 Matematičke formulacije unimodalnih benčmark funkcija 123

16 Karakteristike multimodalnih benčmark funkcija 123

17 Matematičke formulacije multimodalnih benčmark funkcija 124

18 Vrednosti kontrolnih parametara WOA-AEFS metaheuristike 126

19 Komparativna analiza - WOA-AEFS vs. WOA u simulacijama sa benčmark funkcijama bez ograničenja 128
20 Komparativna analiza - WOA-AEFS vs. ostali algoritmi u simulacijama sa benčmark funkcijama bez ograničenja 131
21 CloudSim konfiguracija host računara u simulacijama WOA-AEFS metaheuristike sa pravim skupom podataka 133
22 CloudSim konfiguracija virtuelnih mašina u simulacijama WOA-AEFS metaheuristike sa pravim skupom podataka 134
23 Poboljšanja WOA-AEFS algoritma u odnosu na druge algoritme u simulacijama sa NASA iPSC realnim podacima (vrednosti su iskazane u procentima) 137
24 Komparativna analiza DI indikatora u simulacijama sa NASA iPSC realnim skupom podataka 138
25 Komparativna analiza CPU vremena u simulacijama sa NASA iPSC realnim skupom podataka 139
26 Karakteristike virtuelnih mašina, host računara i data centara u eksperimentima sa veštačkim skupom podataka (WOA-AEFS algoritam) 140
27 Karakteristike poslova u eksperimentima sa veštačkim skupom podataka (WOA-AEFS algoritam) 140

Spisak algoritama

1 Metod Monte Karlo 56
2 Pseudo-kod jednostavnog EA 60
3 Pseudo-kod jednostavnog GA 61
4 Pseudo-kod osnovnog MBO algoritma 70
5 Pseudo-kod WOA metaheuristike 74
6 Pseudo kod MBO-ABC metaheuristike 85
7 Pseudo kod WOA-AEFS 121

1 Uvod

Poslednjih godina, razvojem koncepta klaud računarstva (eng. cloud computing), omogućeno je povezivanje računarskih centara (eng. data centers), koji se fizički nalaze na geografski distribuiranim lokacijama, u jedinstvenu logičku celinu, sa ciljem isporuke kvalitetnih i pravovremenih servisa krajnjim korisnicima. Koncept klaud računarstva se razvio u paradigmu korišćenja računarskih resursa po modelu „plaćanja na osnovu nivoa ostvarene potrošnje resursa"(eng. pay-as-you-go).

Pretragom domaće literature može da se nađe više prevoda engleskog termina „cloud computing". U ovoj tezi, za ovaj engleski termin korišćen je srpski termin „klaud računarstvo". Više o prevodu termina „cloud computing"na srpski jezik može se naći u [10].

S obzirom na složenost koncepta klaud računarstva, ovaj pojam teško može da se „sažme“ u jednu definiciju i zbog toga, u relevantnim izvorima literature, može da se nađe više prihvaćenih definicija ovog pojma. Jednu od najsveobuhvatnijih definicija pojma klaud računarstva dao je Nacionalni Institut za Standarde i Tehnologiju (eng. National Institute for Standards and Technology – NIST) u publikaciji pod oznakom SP 800-145, prema kojoj se klaud računarstvo definiše kao „model koji omogućava pouzdan pristup na zahtev (eng. on-demand) zajedničkim skupu (eng. shared pool) računarskih resursa (mrežama, skladištima podataka, serverima, servisima i aplikacijama) koji se mogu konfigurisati putem računarske mreže, gde se resursi mogu brzo pribaviti i osloboditi, a da su pritom uloženi napor i interakcija davaoca usluge provajdera (eng. service provider) minimalni"[75].

U prilog činjenici da je klaud računarstvo aktuelna i živa multidisciplinarna oblast ide veliki broj objavljenih radova u eminentnim međunarodnim časopisima koji se nalaze na Clarivate Analytics SCI listama, Web of Science (WoS) i Scopus bazama, kao i na ugledmim međunarodnim konferencijama koje organizuju IEEE (Institute of Electrical and Electronics Engineers) i ACM (Association for Computing Machinery). Takođe, na osnovu naučnih rezultata prikazanih u obljavljenim radovima iz ovog domena, zaključuje se da u klaud okruženju postoji veliki broj izazova i problema, za čije rešavanje mogu da se pronađu bolje metode, tehnike i algoritmi. Kao primeri takvih problema i izazova navode se sledeći: sigurnost, raspoređivanje poslova, balansiranje opterećenja, skaliranje, upravljanje kvalitetom usluga (eng. quality of

service - QoS), upravljanje potrošnjom energije računarskih centara, raspoloživost servisa, praćenje performansi, itd.

Svakako, jedan od najvažnijih izazova na klaudu jeste raspoređivanje zahteva krajnjih korisnika (eng. end-users) za izvršavanje na ograničenom skupu raspoloživih virtuelnih mašina (eng. virtual machines - VM). Pojam raspoređivanja se koristi godinama u mnogim aktuelnim istraživačkim oblastima, u kontekstu planiranje poslova i isporuke, raspoređivanje procesa na nivou operativnog sistema, itd. U opštem smislu, problem raspoređivanja odnosi se na definisanje rasporeda izvršavanja zadataka na ograničenom skupu raspoloživih resursa uzimajući pritom u obzir potencijalna ograničenja i funkciju cilja koju je potrebno optimizovati. Glavni cilj raspoređivanja poslova u klaud okruženju jeste kreiranje rasporeda, na osnovu koga se definiše kada i koji virtuelni resursi će obraditi zahteve krajnjih korisnika.

U savremenoj literaturi iz oblasti računarstva, termini „raspoređivanje poslova", "raspoređivanje resursa", „raspoređivanje zadataka"i „raspoređivanje klaudleta"koriste se kao sinonimi. Zbog toga su i u ovoj doktorskoj disertaciji navedeni termini korišćeni kao sinonimi. Potrebno je posebno da se objasni izvor termina klaudlet (eng. cloudlet). Ovaj termin je usvojen iz terminologije koja se koristi u CloudSim simulatoru, gde klaudlet označava zadatak koji je poslat od strane krajnjeg korisnika na izvršavanje u klaud okruženju, i kao takav prihvaćen je u svetskoj literaturi iz ove oblasti.

Pretragom raspoložive stručne literature, zaključuje se da postoje brojne definicije raspoređivanja poslova na klaudu. Prema jednoj definiciji, koju je formulisao Pinedo [91], raspoređivanje je proces donošenja odluka, koji se redovno koristi u mnogim uslužnim i proizvodnim industrijama. Sa ciljem optimizacije jedne ili više funkcija cilja, raspoređivanje se odnosi na alokaciju raspoloživih sistemskih resursa na zadatke koje je potrebno obraditi u određenom vremenskom periodu. U distribuiranim okruženjima poput klauda, glavna uloga komponente raspoređivanja je izbor resursa na kome će se klaudlet izvršiti. Ova odluka zavisi pre svega od funkcije cilja. Jedna od najčešće korišćenih funkcija cilja, koja u značajnoj meri utiče na kvalitet klaud usluga, je vreme završetka svih poslova [16]. Ostale funkcije cilja koje se često koriste, kako u praksi tako i za potrebe simulacija su maksimizacija propusne moći (eng. throughput), balansiranje opterećenja (eng. load balancing) i korisćenja resursa i minimiacija troškova računarskih resursa i potrošnje električne energije.

Takođe, u savremenoj literaturi postoji veliki broj taksonomija algoritama i metoda za raspoređivanje poslova na klaudu. Prema jednoj, češće korišćenoj podeli, algoritmi i tehnike raspoređivanja poslova dele se na statičke i dinamičke algoritme. U slučaju statičkog raspoređivanja, zadaci se raspoređuju za izvršavanje na raspoložive virtuelne mašine pre izvršavanja, tj. tokom samog kompajliranja (eng. compile time) programa za raspoređivanje. Glavni cilj algoritama statičkog raspoređivanja je minimizacija ukupnog vremena izvršavanja svih zadataka (eng. makespan). Tehnike statičkog raspoređivanja nemaju nemogućnost da se dinamički prilagođavaju u zavisnosti od trenutnog opterećenja virtuelnih mašina.

S druge strane, primenom metoda dinamičkog raspoređivanja vrši se preraspodela poslova (procesa) između aktivnih virtuelnih mašina tokom samog izvršavanja (eng. execution time) programa raspoređivanja. Preraspodela se vrši tako što se zadaci sa virtuelnih mašina koje imaju veće opterećenje dinamički prebacuju za izvršavanje na virtuelnim mašinama koje imaju manje opterećenje. U literaturi je dinamičko raspoređivanje poznato i kao balansiranje opterećenja.

Problemi raspoređivanja poslova i balansiranja opterećenja u klaud okruženju spadaju u grupu NP [1] teških (eng. non-deterministic polinomial-time) kombinatornih i/ili globlanih problema sa ili bez ograničenja. Za NP teške izazove relativno lako može da se napiše deterministički algoritam, koji za određeni skup ulaza uvek generiše isti skup izlaza, ali bi za njegovo izvršavanje bilo potrebno mnogo vremena i zbog toga se ovakvi problemi smatraju praktično nerešivim. Iz tih razloga je za njihovo rešavanje potrebno koristiti metaheurističke algoritme i metode. Bez obzira što metaheuristike ne garantuju da će pronaći optimalno rešenje, za većinu NP teških problema metaheuristički algoritmi uspevaju da postignu suboptimalne zadovoljavajuće rezultate u razumnom vremenskom intervalu.

Na osnovu publikovanih rezultata u relevantnim literaturnim izvorima, vidi se da su metaheuristike inteligencije rojeva (eng. swarm intelligence), koje spadaju u grupu prirodom-inspirisanih algoritama, uspešno primenjivana, kako na benčmark, tako i na praktične NP teške optimizacione probleme (globalne i kombinatorne) i da

[1] S obzirom da se u doktorskoj disertaciji spominje veliki broj termina i pojmova iz oblasti računarstva, a posebno nazivi algoritama, koji se u svetskoj literaturi referišu engleskim skraćenicima, u disertaciji su za ovakve pojmove korišćene skraćenice na engleskom jeziku. Ostali, nestandardizovani termini u svetskoj literaturi su uniformno navođeni na srpskom jeziku uz početno navođenje engleskog originala

mogu da postignu bolje rezultate u smislu brzine konvergencije i kvaliteta rešenja, od drugih metoda, tehnika i algoritama.

Problemi raspoređivanja poslova i balansiranja opterećenja u klaud okruženju spadaju u grupu aktivnih i važnih multidisciplinarnih oblasti koje obuhvataju računarstvo, telekomunikacije, elektrotehniku, matematiku i operaciona istraživanja. Na osnovu pretrage dostupne literature, vidi se da su navedeni problemi rešavani primenom različitih metoda, tehnika i metaheuristika. S druge strane, takođe se vidi da ovi problemi nisu dovoljno rešavani primenom metaheuristika inteligencije rojeva.

Činjenice da problemi raspoređivanja poslova i balansiranja opterećenja na klaudu spadaju u kategoriju NP teških problema kombinatorne i/ili globalne optimizacije, sa ili bez ograničenja, kao i da algoritmi rojeva postižu dobre rezultate u rešavanju ovakvih izazova, poslužile su kao osnovni motivator za sprovođenje istraživanja, koje je prikazano u ovoj doktorskoj disertaciji.

1.1 Predmet i cilj istraživanja, hipoteze i naučni doprinos

U skladu sa svim navedenim, predmet istraživanja doktorske disertacije može da se formuliše kao:

- pregled modela raspoređivanja poslova i balansiranja opterećenja u klaud okruženju;
- pregled postojećih metaheurističkih i heurističkih metoda za rešavanje problema raspoređivanja poslova i balansiranja opterećenja u klaud okruženju;
- unapređenje rešavanja problema raspoređivanja poslova i balansiranja opterećenja na klaudu primenom metaheuristika inteligencije rojeva i
- detaljna komparativna analiza navedenih metoda, sa isticanjem prednosti i nedostataka jednih metoda u odnosu na druge.

Opšta hipoteza, koja je poslužila kao polazna osnova za istraživanje koje je prikazano u ovoj doktorskoj disertaciji može da se formuliše kao:

Hipoteza X: Primenom do sada nedovoljno korišćenih metaheuristika inteligencije rojeva za rešavanja problema raspoređivanja poslova i balansiranja opterećenja u klaud okruženju, moguće je generisati bolja rešenja u smislu brzine konvergencije i kvaliteta

(suboptimalna, pravi optimum), čime će se unaprediti rezultati koje su ostvarile druge optimizacione metode za istu instancu problema.

Posebna hipoteza, koja proizilazi iz opšte i koja se odnosi na obrađivanje delova predmeta istraživanja glasi:

X0.1: Moguće je adaptirati i unaprediti postojeće implementacije algoritama rojeva za rešavanje problema raspoređivanja poslova i balansiranja opterećenja u klaud okruženju.

Daljim preciziranjem navedene posebne hipoteze, formulišu se pojedinačne, koje se odnose na elementarne činioce predmeta istarživanja:

X0.1.1: Moguće je adaptirati postojeće implementacije metaheuristika rojeva, koje do sada nisu primenjivane u ovom domenu, za rešavanje problema raspoređivanja poslova i balansiranja opterećenja na klaudu.

X.0.1.2: Moguće je unaprediti postojeće implementacije metaheuristika rojeva, koje su ranije već bile primenjene u ovom domenu, za rešavanje problema raspoređivanja poslova i balansiranja opterećenja u klaud okruženju, podešavanjem kontrolnih parametara, modifikacijama u jednačinama pretrage i/ili hibridizacijama sa drugim algoritmima i metaheuristikama.

Primarni cilj istraživanja u doktorskoj disertaciji je utvrđivanje:

- koliko su problemi raspoređivanja psolova i balansiranja opterećenja u klaud okruženju optimizovani sa primenjenim metodama i tehnikama optimizacije;
- prednosti i nedostaci primenjenih metoda i metaheuristika za rešavanje ovih problema i
- unapređenje rešavanja problema raspoređivanja poslova i balansiranja opterećenja u klaud okruženju, do koga se dolazi primenom algoritama inteligencije rojeva.

S obzirom da je tokom sprovodenog istraživanja za potrebe ove doktorske disertacije implementirano i adaptirano više osnovnih i unapređenih algoritama inteligencije rojeva, sekundarni cilj istraživanja prikazanog u ovoj doktorskoj tezi jeste unapređenje postojećih implementacija metaheuristika inteligencije rojeva minornim (promena jednačine pretrage, kontrolnih parametara, itd.) i/ili krupnim modifikacijama (hibridizacijom sa drugim metaheuristikama i heuristikama).

Glavna metoda koja je korišćenja u istraživanju je simulacija sa standardnim skupovima podataka i u okviru standardnog okruženja. U simulacijama su korišćeni podaci iz realnog klaud okruženja, ali i veštački podaci koji su generisani u simulatoru. Rezultati su potvrđeni i unapređenja su verifikovana komparativnim analizama sa rezultatima koje su generisale druge primenjene metode i čiji su rezultati publikovani u vrhunskim međunarodnim časopisima i u zbornicima međunarodnih konferencija.

Glavni naučni doprinos istraživanja koje je prikazano u doktorskoj disertaciji je:

- unapređenje rešavanja problema raspoređivanja poslova i balansiranja opterećenja u klaud okruženju primenom i adaptacijom algoritama inteligencije rojeva, kako osnovnih, tako i unapređenih verzija;
- unapređenje postojećih verzija i implementacija metaheuristika rojeva;
- kritički osvrt na rezultate do kojih dolaze druge metode i tehnike za rešavanje problema raspoređivanja poslova i balansiranja opterećenja na klaudu;
- komparativna analiza rezultata raznih algoritama inteligencije rojeva za ovu klasu problema, kao i između algoritama inteligencije rojeva i drugih metoda i algoritama i
- detaljan prikaz matematičkih modela i formulisanja problema raspoređivanja poslova i balansiranja opterećenja u klaud okruženju.

Rezultati istraživanja koji su prikazani u ovoj doktorskoj disertaciji validirani su od strane svetske naučne zajednice objavljivanjem u jednom vrhunskom i jednom međunarodnom časopisu od nacionalnog značaja referisanog u Scopus bazi, poglavlju Springer knjige iz serijala LNCS (Lecture Notes in Computer Science) i u zborniku jednog od svetskih vodećih kongresa iz domena evolutivnog računarstva:

- **Ivana Strumberger**, Nebojsa Bacanin, Milan Tuba, Eva Tuba, *Resource Scheduling in Cloud Computing Based on a Hybridized Whale Optimization Algorithm*, Applied Sciences, Vol. 9, Issue 22, MDPI, 2019, pp. 4893, doi: https://doi.org/10.3390/app9224893 (kategorija M22) [114];
- **Ivana Strumberger**, Milan Tuba, Nebojsa Bacanin, Eva Tuba, *Cloudlet scheduling by hybridized monarch butterfly optimization algorithm*, Vol. 8, Issue 3, MDPI, 2019, pp. 44, doi: https://doi.org/10.3390/jsan8030044 (kategorija M51) [121];

- Nebojsa Bacanin, Eva Tuba, Timea Bezdan, **Ivana Strumberger**, Milan Tuba, *Artificial Flora Optimization Algorithm for Task Scheduling in Cloud Computing Environment*, In: Yin H., Camacho D., Tino P., Tallón-Ballesteros A., Menezes R., Allmendinger R. (eds) Intelligent Data Engineering and Automated Learning – IDEAL 2019. IDEAL 2019. Lecture Notes in Computer Science, vol 11871. Springer, Cham, doi: https://doi.org/10.1007/978-3-030-33607-3_47 (kategorija M13) [11] i

- **Ivana Strumberger**, Eva Tuba, Nebojsa Bacanin, Milan Tuba, *Dynamic tree growth algorithm for load scheduling in cloud environments*, Proceedings of the IEEE Congress on Evolutionary Computation 2019 (CEC 2019), Wellington, New Zealand, 10-13 June, 2019, pp. 65-72, doi: 10.1109/CEC.2019.8790014 (kategorija M33) [110].

Tokom istraživanja implementirane su i adaptirane osnovne i unapređenje/hibridizovane verzije četiri metaheuristike inteligencije rojeva za rešavanje problema raspoređivanja poslova i balansiranja opterećenja na klaudu. Zbog većeg značaja, u praktičnim (eksperimentalnim) poglavljima doktorske teze detaljno su prikazani rezultati koji su verifikovani publikovanjem u međunarodnim časopisima.

1.2 Struktura doktorske disertacije

Doktorska disertacija se sastoji iz uvoda, sedam glava, zaključka i korišćene literature.

Nakon uvoda, u prvoj glavi prikazani su pojam i osnovni koncepta klaud računarstva. Na početku ove glave navedeni su razlozi zbog kojih se paradigma klaud računarstva sve češće koristi, kako od strane individualnih, tako i od strane korporativnih korisnika. Zatim su opisane tehnologija virtuelizacije i hiper-konvergirane infrastrukture, kao osnovne tehnologije koje omogućavaju sam koncept klaud računarstva. U okviru posebnog poglavlja u ovoj glavi date su najvažnije definicije klaud računarstva, koje su prihvaćene u svetskoj literaturi, zajedno sa tri osnovna modela usluga i četiri modela isporuke. Konačno, na kraju prve glave ukratko je prikazana arhitektura savremenih sistema klaud računarstva.

Obzirom da glavni problem koji je razmatran u ovoj disertaciji pripada grupi

optimizacionih problema, u drugoj glavi definisan je pojam i data je osnovna taksonomija optimizacionih problema. Optimizacioni problemi su razvrstani na osnovu tri kriterijuma: tipu varijabli, na osnovu ograničenja i prema broju funkcija cilja. Na kraju ove glave ukratko su opisani optimizacioni algoritmi i uvedene su klase kompleksnosti.

Treća glava je posvećena modelima raspoređivanja poslova i balansiranje opterećenja u klaud računarstvu. Na početku glave je data osnovna terminologija i ukratko su prikazani algoritmi koji se najčešće koriste za raspoređivanje poslova na klaudu. Nakon toga opisan je prvi model koji je korišćen u praktičnim simulacijama i koji spada u grupu optimizacionih problema sa jednom funkcijom cilja, gde je uzeta u obzir minimizacija ukupnog vremena izvršavanja svih poslova na raspoloživim klaud računarskim resursima. U ovoj glavi takođe je prikazan i drugi model, čije je rešavanje prikazano u eksperimentalnim poglavljima disertacije, a koji pripada kategoriji višekriterijumske optimizacije sa ograničenjima, gde su uzete u obzir dve funkcije cilja: ukupno vremene izvršavanja svih poslova i minimizacija troškova u okviru raspoloživog budžeta.

U četvrtoj glavi je dat pregled i jedna od najšire korišćenih klasifikacija optimizacionih algoritama. Takođe su prikazane heurističke metode optimizacije. Zbog relevantnosti sa istraživanjem sprovedenog za potrebe ove disertacije, posebna pažnja je posvećena metaheuristikama i klasifikaciji na metaheuristike koje su inspirisane prirodom i na metaheuristike koje nisu inspirisane prirodom.

Cela peta glava posvećena je metaheuristikama inteligencije rojeva. Prikazani su osnovni principi i data je definicija algoritama rojeva, kao i dve najvažnije komponente procesa pretrage: eksploatacija (intenzifikacija) i eksploracija (diversifikacija). Nakon toga, detaljno su prikazane osnovne verzije algoritama koji su unapređeni hibridizacijom i adaptirani za rešavanje problema raspoređivanja poslova i balansiranja opterećenja na klaudu - algoritam optimizacije monarh leptirovima i optimizacija jatom kitova. Na kraju ove glave dat je pregled literature, gde su prikazane objavljene implementacije metaheuristika rojeva za NP teške probleme, uključujući i problem koji je razmatran u ovoj disertaciji.

Šesta i sedma glava su eksperimentalne glave, u kojima su detaljno prikazane praktične simulacije i rezltati do kojih se došlo u sprovedenom istraživanju. U šestoj

glavi detaljno je prikazan unapređeni hibridizovani algoritam optimizacije monarh leptirovima, dok je u sedmoj glavi opisan poboljšani hibridni pristup optimizacije jatom kitova. U oba slučaja, prvo su prikazani nedostaci osnovnih metaheuristika i podešavanja kontrolnih parametara. Zatim su detaljno prikazani i analizirani rezultati koje su obe metaheuristike ostvarile u simulacijama sa globalnim funkcijama bez ograničenja, gde je prikazana i komparativna analiza sa drugim vrhunskim algoritmima čiji su rezultati objavljeni u modernoj literaturi. Na kraju su dati svi detalji eksperimenata rešavanja problema raspoređivanja poslova i balansiranja opterećenja na klaudu. Za svaki algoritam su rađena dva tipa eksperimenata - jedan sa veštačkim podacima,generisanim u simulatoru i drugi sa podacima iz realnog, produkcionog klaud okruženja, koji su preuzeti iz javnih dostupnih benčmark baza. U slučaju oba algoritma detaljno je prikazano okruženje koje je korišćeno u simulacijama i data je komparativna analiza sa drugim vrhunskim heurističkim i metaheurističkim metodama, koje su adaptirane i validirane na istim instancama problema i pod istim eksperimentalnim uslovima.

U zaključku je dat rezime sprovedenog istraživanja sa taksativno navedenim ostvarenim rezultatima i istaknutim naučnim doprinosom. Takođe, u ovom delu disertacije prikazani su potencijalni budući istraživački pravci u domenu primena metaheuristika inteligencije rojeva na probleme i izazove klaud računarstva.

2 Pojam i koncept klaud računarstva

Klaud računarstvo spada u grupu novijih računarskih paradigmi, koja se kao i paradigma mrežnog računarstva (eng. grid computing) oslanja na grupisanje resursa u grupe (eng. resource pooling) i na upotrebu mrežnih i Internet tehnologija. Mnogi vide klaud računarstvo kao sledeći korak u razvoju mrežnog računarstva. Međutim, između ove dve paradigme postoje značajne razlike. Dok je jedan od fokusa klaud računarstva eliminisanje potreba za kupovinu hardvera i softvera alociranjem resursa između različitih servera grupisanih u klastere (eng. server clusters), mrežno računarstvo je računarska tehnologija koja kombinuje računarske resurse iz različitih domena sa ciljem ostvarivanja zajedničkog zadatka.

Klaud računarstvo se pre svega odnosi na novi način isporuke računarskih resursa, gde se računarski resursi iznajmljuju kao usluge. Pod računarskim resursima podrazumeva se gotovo sve, od podataka i softvera, do hardverskih komponenti, poput procesorskog vremena, memorije, skladišta, itd. Korišćenjem koncepta klaud računarstva, kompanije više ne moraju da imaju velike početne troškove prilikom kupovine opreme i softvera, što u većini slučajeva predstavlja kapitalne investicije, već sve to mogu da iznajmljuju od entiteta koji se naziva provajder klaud usluga (eng. cloud service provider - CSP).

Ideja iza klaud računarstva je jednostavna i inspirisana je električnom distribucionom mrežom, gde se električna energija kupuje u vidu usluge. U klasičnom sistemu distribucije električne energije, kupci (korisnici) plaćaju samo za onoliko energije koliko su i potrošili. S druge strane, kupcima nije važno odakle (sa koje lokacije) se energija doprema. Jedino što je važno jeste da je dopremanje energije permanentno i da je energija određenog kvaliteta (u ovom slučaju snage). Slično je i sa klaud računarstvom. Sve računarske i softverske komponente se krajnjim korisnicima dopremaju putem širokopojasne mreže, kao što je Internet, gde korisnici plaćaju za samo onoliko resursa koliko su potrošili i pritom korisnicima nije važno odakle (u ovom slučaju sa kog servera, ili klastera servera) su računarski resurs dobili. Korisnicima je jedino važno da su svoj resurs dobili pravovremeno i da je resurs određenog kvaliteta (performansi).

U veku brzog razvoja mrežnog računarstva i komunikacionih tehnologija, klaud računarstvo, kao paradigma koja tek nastaje, demonstrira kako upotreba pravih

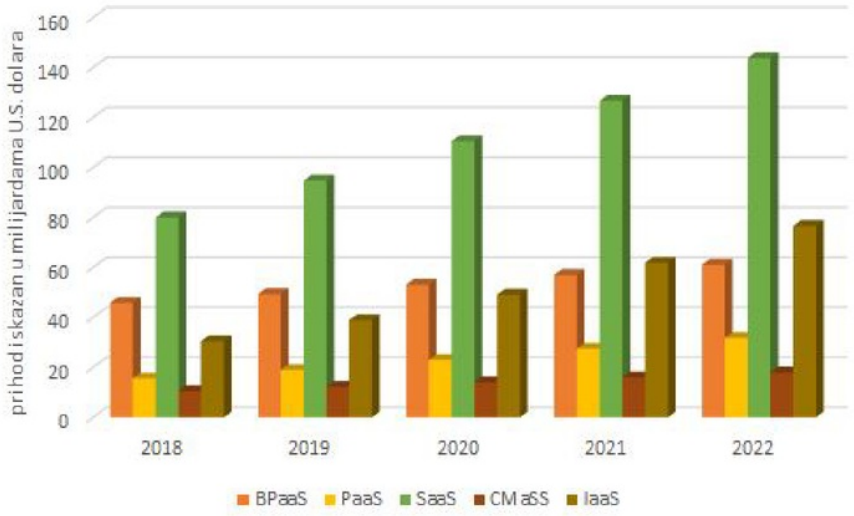
resursa i dostupne snage za obradu podataka omogućava infrastrukturu koja obezbeđuje i olakšava adekvatnu isporuku hardverskih i softverskih resursa preko mreže, sa glavnim ciljem da zadovolji sve složenije potrebe i zahteve krajnjih korisnika. Zbog svoje efikasnosti u pružanju usluge isporuke računarskih resursa, klaud računarstvo je široko usvojeno u industriji informacionih tehnologija (IT), kao i u akademskim i istraživačkim zajednicama širom sveta.

Klaud računarska infrastruktura se organizuje u formi klaud računskih centara (eng. cloud data centers). U svakom klaud računskom centru nalazi se određeni broj fizičkih servera, koji se u stručnoj literaturi najčešće nazivaju host računari (eng. hosts). Svaki host računar određuju karakteristike (atributi), kao što su jedinstveni identifikator hosta (eng. host unique identifier - hostID), broj raspoloživih procesirajućih elemenata (eng. processing elements - PE), performanse svakog procesirajućeg elementa, raspoloživi mrežni propusni opseg (eng. bandwidth), itd. Dalje, obzirom da je virtuelizacija jedna od vodećih tehnologija koja je omogućila sam koncept klaud računarstva, na svakom fizičkom serveru (hostu) nalazi se više virtuelnim mašina koje primenjuju politiku (polisu) korišćenja resursa fizičkog računara koja se bazira na vremenu (eng. time-shared policy) ili na prostoru (eng. space-shared policy).

Kada krajnji korisnici pošalju svoje zahteve (poslove) na klaud, zahteve prvo prihvata komponenta koja se u literaturi najčešće naziva menadžer zadataka (eng. task manager) i koja organizuje dolazne zadatke (zahteve) i šalje informaciju o statusu svakog zadatka korisniku koji ga je poslao. Menadžer zadataka zatim prosleđuje zahteve raspoređivaču poslova (eng. task scheduler), koji dodeljuje (mapira) zadatke na raspoložive i odgovarajuće (u smislu hardverskih karakterisika) virtuelne mašine korišćenjem algoritma za raspoređivanje poslova (eng. task scheduling algorithm). U slučaju da u datom trenutku ni jedna virtuelna mašina nije raspoloživa, zadatak će čekati u redu čekanja (eng. task queue) dok se potrebni resursi (virtuelne mašine) za obradu tog zadatka ne oslobode. Nakon što virtuelna mašina završi sa obradom dodeljenog zadatka, virtuelna mašina se oslobađa i spremna je za prihvanje sledećeg zadatka.

2.1 Motivacija i potreba za klaud računarstvom

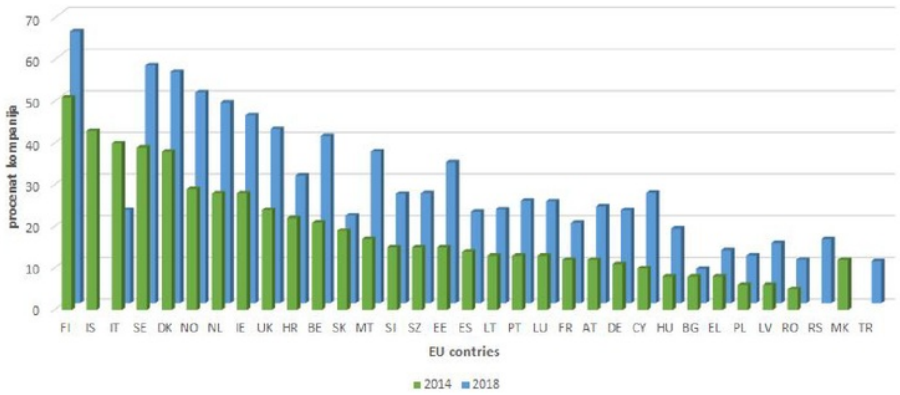
Tržište klaud računarstva, kao i prihvatanje ovog modela, kako u malim, tako i u srednjim i velikim organizacijama brzo raste. Prema Gartner-u, ukupna vrednost usluga javnog klauda je iznosile 182,4 milijarde američkih dolara u 2018. godini, uz predviđanje da će njegova vrednost dostići 214,3 milijarde američkih dolara do kraja 2019. godine [2]. Pretpostavke kompanije Gartner o veličini svetskog tržišta javnog klauda do 2022 godine (Slika 1 [114] - podaci koji su korišćeni za generisanje grafikona su preuzeti iz [2]) demonstriraju značaj i uticaj klaud računarstva i klaud usluga na krajnje korisnike i kompanije [2].



Slika 1: *Predviđanje vrednosti globalnog javnog klaud tržišta prema kompaniji Gartner; BpaaS—business process as a service; CMaSS—cloud management and society services.*

Pored toga, tržište klaud računarstva i usvajanje ovog koncepta naglo raste i u Evropi, gde je prema studijama Eurostata, tokom 2018. godine 26% organizacija koristilo klaud računarstvo za e-usluge i skladištenje e-dokumenata, dok je 55% njih koristilo neke od naprednih usluga sa klauda, kao što su finansijske i računovodstvene aplikacije i sistemi za upravljanje odnosa sa klijentima (eng. customer relationship management - CRM). Takođe, 18% više organizacija je koristilo usluge javnog klaud računarstva u poređenju sa privatnim klaudom [1]. Na Slici 2 [114] (podaci koji su korišćeni za generisanje grafikona su preuzeti iz [1]) prikazan je procenat kompanija

iz Evropske Unije (EU) koje su tokom 2014. i 2018. godine koristile neku od klaud usluga.



Slika 2: *Procenat EU kompanija koji je koristio neku od klaud usluga tokom 2014. i 2018. godine.*

Da bi se opravdala motivacija i potreba za klaud računarstvom, važno je da se pomenu i razumeju razlozi koji su uticali na njegovu široku primenu i popularnost, kao i faktori koji su uticali na nameru da se koristi ovaj relativno nov model isporuke računarskih resursa, pre svega od strane organizacionih korisnika. Sveoubhvatna studija koja pokazuje razloge za usvjanje i prepreke za korišćenje klaud računarstva u malim i srednjim organizacijama se može naći u [87]. Prema ovoj studiji, ušteda troškova i pojednostavljenje tehnološke infrastrukture predstavaljju dva najvažnija faktora koja su uticala na veću upotrebu koncepta klaud računarstva. Prema istoj studiji, nedostatak standardizacije, potreba za IT profesionalcima i faktori nepoverenja su samo neke od prepreka za usvajanje koncepta klaud računarstva [87].

Glavna motivacija koja stoji iza brzog rasta upotrebe klaud računarstva je novo generisana vrednost, posebno za domene poslovanja, ekonomiju i računarstvo. Prema [88], neke od važnijih predosti koje se generišu isporukom računarskih resursa u vidu usluga su smanjenje troškova softvera i hardvera, smanjenje troškova održavanja infrastrukture i operativnih torškova, pristup uslugama sa bilo kog mesta na svetu u bilo koje vreme po principu 24/7, potencijal za transformaciju poslovnog procesa, jednostavna rešenja koja se koriste na klaudu, itd.

Zahvaljujući predostima tehnologije virtuelizaije, efikasnost i stepen iskorišćeno-

sti raspoloživih računarskih resursa značajno se povećavaju, što dovodi do daljeg smanjenja, kako operativnih, tako i dugoročnih troškova. Za razliku od klaud okruženja, tradicionalna IT infrastruktura, gde kompanije sve svoje računarske resurse implementiraju u svojim prostorijama (eng. on premises), organizovana je formi tzv. „IT silosa". IT silosi funkcionišu kao zasebne jedinice i često nisu komunikaciono povezani na adekvatan način, što može za posledicu da ima postojanje suvišnog i nepotrebnog hardvera i suboptimalno korišćenje računarskih resursa. U takvim okruženjima je vrlo teško smanjiti računarsku infrastrukturu. S druge strane, klaud okruženje omogućava da se efikasno i brzo povećaju ili smanje IT odeljenja u skladu sa poslovnim potrebama i zahtevima. Više o radicionalnom načinu organizovanja IT resursa može se naći u [10].

U savremenom Internet dobu, uz sve učestalije i intenzivnije sajber napade, sigurnost podataka i sistema postaje jedno od prioritetnih zahteva. Organizacije u većini slučajeva finansijski ne mogu da priušte kupovinu sofisticiranih bezbednosnih uređaja (eng. security appliances), dok su istovremeno njihovi sistemi i podaci izloženi velikim rizikom. Međutim, klaud provajderi nude korisnicima najsavremenije sigurnosne uređaje koji su konfigurisani i sposobni da spreče razne tipove sajber napada, dok istovremeno omogućavaju slojevitu zaštitu podataka, resursa i servisa u klaud okruženju. Klaud provajderi implementiraju bezbednosne mehanizme na više nivoa, od klaud firewall-a, sve do nivoa virtuelnih mašina. Ovakav tip zaštite u domenu računarske bezbednosti poznat je slojevita sigurnost (eng. layered security), ili duboka odbrana (eng. defense in depth).

Upotrebom koncepta i modela klaud računarstva i kombinacijom svih gore navedenih prednosti klaud okruženja, krajnji korisnici i kompanije mogu da imaju velike potencijalne koristi ako se odluče da sa tradicionalnog načina organizovanja IT resursa usvoje paradigmu klaud računarstva.

2.2 Tehnologija virtuelizacije i hiper-konvergirana infrastruktura

Dve osnovne tehnologije koje su omogućile realizaciju koncepta klaud računarstva uz visok nivo raspoloživosti i fleksibilno upravljanje resursima su virtuelizacija (eng. virtualization technology) i hiper-konvergirana infrastruktura (eng. hyper-converged

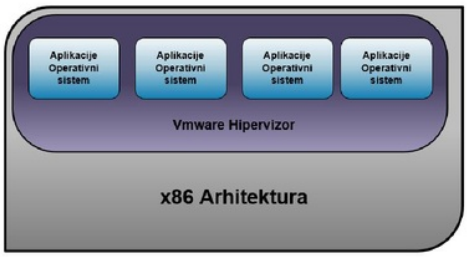
infrastructure - HCL). Kako bi se uspešno izvršili primljeni zahtevi, klaud okruženje je podržano sa ovim tehnologijama. Bez virtuelizacije i hiper-konvergirane infrastrukture, klaud računarstvo kao model isporuke računarskih resursa u vidu usluga, ne bi bilo ekonomski opravdano, kako za klaud klijente (krajnje korisnike), tako ni za klaud provajdere.

Prema jednoj od najjednostavnijih definicija, koja uzima u obzir najčešći razlog za korišćenje tehnologije virtuelizacije, virtuelizacija je tehnologija koja omogućava pokretanje više instanci operativnih sistema u okviru jednog, tzv. operativnog sistema domaćina (eng. host operating system). Prema drugoj definiciji, koja stavlja naglasak na upotrebu virtuelizacije od strane korporativnih korisnika, tehnologija virtuelizacije omogućava istrovremeno izvršavanje različitih zadataka od strane virtuelnih mašina, koje se nalaze iznad zajedničke hardverske fizičke infrastrukture, ali se svaka mašina izvršava u svom izolovanom okruženju (eng. sandbox) [44]. U opštem kontektstu, virtuelizacija je tehnologija koja apstrahuje računarske resurse tako što emulira fizičke resurse pomoću softvera.

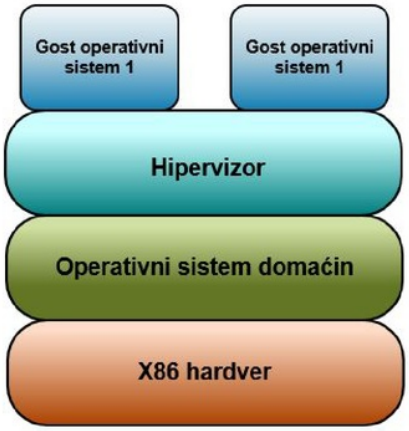
Hipervizor, ili menadžer virtuelnih mašina (eng. virtual machine manager - VMM), predstavlja softversku komponentu koja izvodi upravljačke i kontrolne operacije virtuelnog okruženja tako što upravlja procesima kao što su kreiranje, komunikacija virtuelnih komponenti sa fizičkim, koordinacija rada i uništavanje virtuelnih mašina. Hipervizor je u savremenoj literaturi poznat još i pod nazivom monitor virtuelnih mašina (eng. virtual machine monitor - VMM).

U praksi su poznata dva tipa hipervizora - tip 1 i tip 2. Hipervizor tip 2, koji je poznat pod nazivom ogoljeni hipervizor (eng. bare-metal hypervisor), izvršava se direktno iznad fizičkog hardvera i ima ulogu operativnog sistema za virtuelizaciju. S druge strane, tip 2 hipervizor se izvršava iznad klasičnog operativnog sistema računara i u literaturi je zbog toga poznat pod nazivom ugošćeni hipervizor (eng. hosted hypervisor). U terminologiju hipervizora tip 2, kako bi se napravila razlika između operativnog sistema koji se izvršava neposredno iznad fizičkog hardvera (iznad kojega se izvršava tip 2 hipervizor) i operativnog sistema koji se izvršava iznad (u okviru) virtuelne mašine, koriste se termini „host"i „guest", gde „host"označava glavni operativni sistema, dok se termin „guest"odnosi na operativni sistem koji se izvršava iznad virtuelne mašine.

Hipervizor tip 1 je mnogo brži i efikasniji od hipervizora tip 2, jer eliminiše dodatni sloj u vidu klasičnog operativnog sistema. Zbog toga se tip 1 hipervizor koristi na infrastrukturi klaud provajdera. Kao neki od poznatijih predstavnika hipervizora tip 1 navode se VMware ESXi, Oracle VM Server for x86 i Microsoft Hyper-V, dok se predstavnici hipervizora tip 2, poput Oracle VirtualBox, VMware Workstation, VMware Player i Parallels Desktop for Mac, uglavnom koriste u manjim korporativnim, ili u kućnim okruženjima. Slojevita arhitektura tip 1 i tip 2 hipervizora prikazana je na slikama 3 i 4 (slike preuzete iz [10]), respektivno.



Slika 3: *Slojevita struktura tip 1 hipervizora*



Slika 4: *Slojevita struktura tip 2 hipervizora*

Kao što je već navedeno, druga tehnologija koja je omogućila paradigmu klaud računarstva je hiper-konvergirana infrastruktura. Ova arhitektura, koja je nastala relativno skoro, je računarska infrastruktura koja virtuelizuje komponente tradicionalne fizičke hardverske strukture korišćenjem koncepta „definisano softverom". Ovakva platforma obuhvata hipervizora, softverski definisanu mrežu (eng. software-defined

network - SDN) i softverski definisano skladište (eng. software-defined storage - SDS).

Upotrebom tehnologija virtuelizacije i hiper-konvergirane infrastrukture, klaud provajder može da isporuči softverske i hardverske resurse u vidu usluge krajnjim korisnicima na efikasan i ekonomičan način putem široko-pojasne komunikacione mreže. Primena ovih tehnologija je takođe omogućila neke od najvažnijih karakteristika klaud računarstva, među kojima su skalabilnost zadataka i dostupnost resursa (eng. resource availability), dinamičko obezbeđivanje (eng. dynamic provisioning), balansiranje opterećenja (eng. load balancing), otpornost na greške (eng. fault tolerance) i interoperabilnost heterogenih računarskih resursa (eng. resource interoperability) [19].

2.3 Definisanje klaud računarstva i osnovni modeli

U literaturi mogu da se nađu mnoge definicije koje objašnjavaju paradigmu klaud računarstva, a prema jednoj od njih navodi se da klaud računarstvo predstavlja elastičan i distribuiran sistem u kome se računarski resursi, kao što su jedinice za obradu podataka, skladišni prostor, informacije i softver, šalju i distribuiraju putem računarske mreže i isporučuju na distribuiranoj lokaciji u klaudu, gde mogu dalje da se dele i preuzimaju [24].

Jedna druga definicija klaud sistem opisuje kao grupu (skup) heterogenih hardverskih i softverskih resursa, koji omogućavaju pružanje usluga krajnjim korisnicima, što implicira da nije neophodno da krajnji korisnici usluga klaud računarstva poseduju sopstvenu hardversku i softversku infrastrukturu [63].

Međutim, bez obzira na dostupnost brojnih definicija pojma klaud računarstva, jednu od najznačajnijih definicija, koju su prihvatili mnogi globalni dobavljači klaud usluga, dao je Nacionalni Institut za Standarde i Tehnologiju (eng. National Institute of Standards and Technology - NIST). Prema NIST-a, klaud računarstvo se definiše kao [75]: „model koji omogućava univerzalan (sa bilo kojeg uređaja) i pouzdan pristup na zahtev (eng. on-demand) zajedničkom skupu (eng. shared pool) podesivih (koji mogu da se konfigurišu) računarskih resursa putem računarske mreže, gde resursi mog brzo da se pribave (eng. provision) i oslobode (eng. release) uz minimalnu interakciju i napor od strane klaud klijenta i klaud provajdera". Dalje, prema organizaciji NIST, suština modela klaud računarstva nalazi se u pet osnovnih karakteristika (koje

proističu iz definicije), tri modela usluga i četiri modela implementacije (isporuke). U literaturi je ovo poznato pod nazivom „osnovnih 5-4-3 principa klaud računarstva"[75].

Pet osnovnih karakteristika klaud računarstva mogu da se sumiraju na sledeći način [75]:

1. samousluživanje na zahtev (eng. on-demand self-service). Klaud klijent može sam da obezbedi mogućnosti, poput serverskog vremena ili mrežnog skladišta kada se za to ukaže potreba, bez interakcije čoveka;

2. široko-pojasni mrežni pristup (eng. broad network access). Svim mogućnostima može da se pristupi preko široko-pojasne mreže poput Interneta, korišćenjem standardnih mehanizama i procedura, gde se za pristup koriste heterogene tanke i debele klijentske platforme, poput lap top računara, pametnih telefona, tablet računara i drugih uređaja;

3. elastični skupovi računarskih resursa (eng. elastic resource pooling). Svi računarski resursi klaud provajdera se grupišu u logičke skupove sa ciljem efikasnog usluživanja kranjih korisnika korišćenjem modela višestrukog zakupa (eng. multitenant model), gde se heterogeni fizički i virtuelni resursi dinamički dodedljuju i oduzimaju u zavisnosti trenutnih potreba klaud klijenata;

4. brza elastičnost (eng. rapid elasticity). Mogućnosti mogu brzo i elastično da se obezbede (u pojedinim slučajevima automatski) sa ciljem brzog skaliranja naviše, ali isto tako mogu i brzo da se oslobode, sa ciljem brzog skaliranja na dole i to sve u zavisnosti od trenutnih potreba krajnjih korisnika i

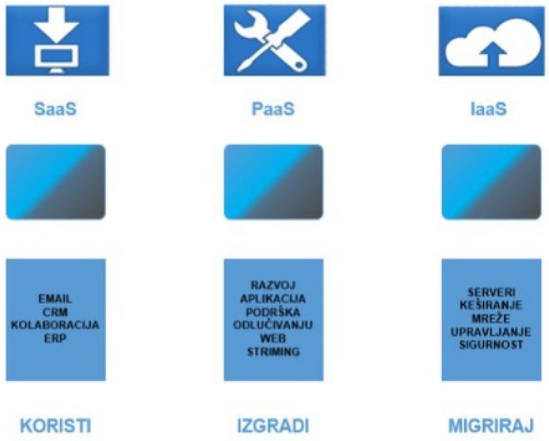
5. merenje potrošnje klaud servisa (eng. measured services). Korišćenje resursa od strane klaud klijenata se nadgleda i kontroliše, na osnovu čega se generiše izveštaj o potrošnji, koji je transparentan, kako za klaud provajdera, tako i za klaud klijenta koji servise koristi.

Na osnovu NIST-ovog viđenja klaud računarstva, nastale su i dve najznačajnije taksonomije ove računarske paradigme, koje su široko prihvaćene u svetskoj literaturi. Prva kategorizacija deli klaud računarstvo na osnovu kriterijuma servisnih modela (model usluga), gde mogu da se izdvoje sledeće tri osnovne grupe modela usluga: softver kao usluga (eng. software as a service - SaaS), platforma kao usluga (eng.

platform as a service - PaaS) i infrastruktura kao usluga (eng. infrastructure as a service - IaaS).

Model SaaS isporučuje aplikacije koje su dostupne klijentima sa strane klaud provajdera u formi „hostovanog" softvera. Model PaaS obezbeđuje računarsku platformu (operativni sistem i razvojna okruženja), na kojoj korisnici klaud usluga mogu da razvijaju, implementiranju, instaliraju i konfigurišu sopstvene aplikacije, dok model IaaS obezbeđuje celu računarsku infrastrukturu, kao što su jedinice za obradu podataka, uređaji za umrežavanje i skladištenje, krajnjim korisnicima u vidu klaud usluga.

Na Slici 5 (slika preuzeta iz [10]) mogu se videti tri osnovne grupe modela klaud računarstva.



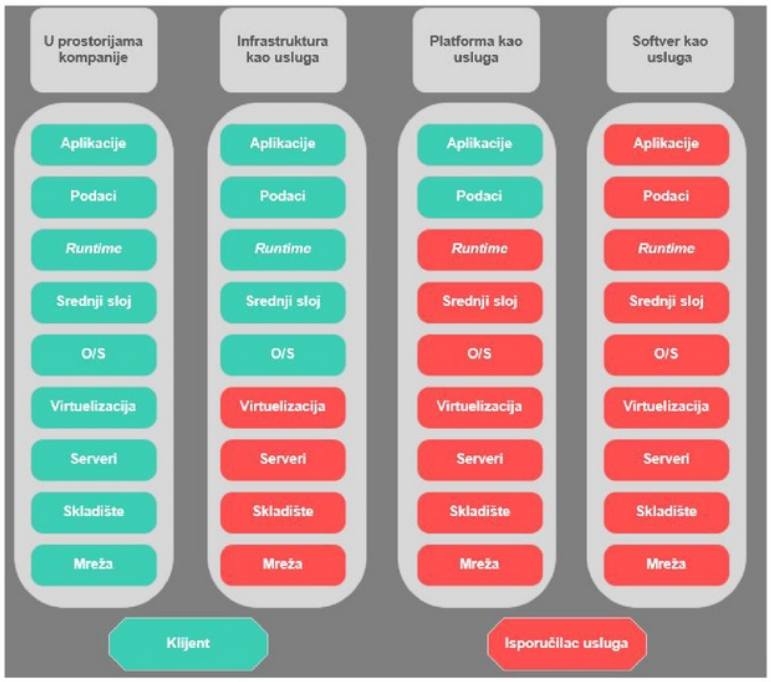
Slika 5: *Tri osnovne grupe klaud računarstva prema tipu usluga*

U praksi se između navedena tri osnovna modela mogu naći i tzv. međumodeli. Tako na primer, neretko se sreću termini poput poslovnog procesa kao usluge (eng. business process as a service - BPaaS), skladišta kao usluge (eng. storage as a service - STaaS) i bilo čega kao usluge (eng. anything as a service - XaaS).

Kada se govori o taksonomiji klaud računarstva prema tipu usluga, potrebno je da se spomene i distribucija odgovornosti (eng. cloud service responsibilities) i kontrole (eng. cloud service control) između klaud provajdera i klaud klijenta. S jedne strane, u slučaju kada korporativni klijent koristi tradicionalnu IT infrastrukturu, svi računarski resursi se nalaze u prostorijama (eng. on-premises) i u vlasništvu kompanije, gde kompanija ima potpuno kontrolu i odgovornost za sve resurse. Kako

korporativni klijent počinje da koristi klaud usluge, u zavisnosti od tipa usluge, transformiše se nivo odgovornosti i kontrole. U slučaju modela SaaS, klaud provajder ima najveći stepen nivoa odgovornosti i kontrole. Kako se od modela SaaS kreće ka modelu IaaS, tako stepen nivoa odgovoronosti i kontrole klaud provajdera opada, dok kluad klijenta raste.

Na Slici 6 (slika preuzeta iz [10]), gde je prikazana distribucija nivoa kontrole i odgovornosti između klaud provajdera i klijenta, svetlo zelenom bojom označene su komponente računarske infrastrukture za koje je odgovoran klaud klijent, dok crvena boja označava slojeve za koje je odgovoran i koje kontroliše klaud provajder.



Slika 6: *Stepen odgovornosti i kontrole nad klaud resursima*

Druga taksonomija klaud računarstva, na osnovu kriterijuma modela implementacije (isporuke), pravi razliku između javnog (eng. public cloud services), privatnog (eng. private cloud services), zajedničkog (eng. community cloud services) i hibridnog klauda (eng. hybrid cloud services). Prema ovoj taksonomiji, akcenat nije na vlasništvu klaud resursa, već na strani koja ih eksploatiše. Na primer, u privatim klaud okruženjima, ceo klaud sistem ekskluzivno koristi jedan entitet, koji može da bude individualni ili korporativni korisnik, dok s druge strane vlasnik klaud resursa

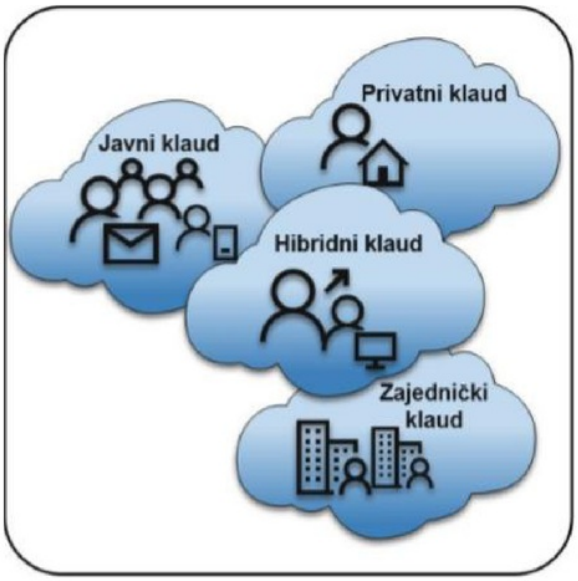
može da bude ili entitet koji ih eksploatiše, ili klaud provajder.

Glavna karakteristika javnog klauda jeste da se javna klaud infrastruktura deli između više korisnika (strana), gde se koriscnici ne poznaju. Drugim rečima, korisnik javnog klaud servisa nema informaciju o broju drugih korisnika koji eksploatišu isti klaud resurs, niti ima bilo kakve informacije o njima. Servisi javnog klauda su najčešći i obuhvataju servise poput IaaS, PaaS i SaaS. Jedan od glavnih izazova javnog klauda je pitanje bezbednosti računarskih resursa.

Zajednički klaud eksploatišu najčešće korporativni korisnici koji se međusobno poznaju i koji imaju sličnu viziju, misiju i ciljeve. U industriji je čest slučaj da se više industrijskih entiteta, koji imaju povezane poslovne procese, udruže i implementiraju zajedničko klaud okruženje.

Konačno, hibridini klaud, koji predstavlja kombinaciju privatnog i javnog klaud okruženja, se najčešće koristi kada organizacija zbog pitanja bezbednosti ne želi da svoje ključne aplikacije (eng. mission critical applications) i servise koristi sa klauda, a s druge strane želi da ostvari uštede. U tom slučaju, kompanije implementiraju privatni klaud, na kojem konfigurišu važne aplikacije i skladišta podataka gde se nalaze osetljivi (poverljivi) podaci, dok druge servise koriste u vidu usluga sa javnog klauda.

Na Slici 7 (slika preuzeta iz [10] prikazani su tipovi klaud računarstva prema kriterijumu modela isporuke.



Slika 7: *Tipovi klauda na osnovu kriterijuma modela isporuke*

Kao neke od poznatijih platformi klaud provajdera navode se Amazon AWS (Amazon Web Services) sa servisima poput EC2 (elastic compute cloud) i S3 (simple storage service), Microsoft Azure, Google GCE (Google Compute Engine), Oracle Cloud i IBM Cloud.

Detaljnije o prednostima i nedostacima svakog od navedenog modela usluge i isporuke klaud računarstva može da se pronađe u [10].

2.4 Arhitektura klaud računarstva

Arhitektura klaud računarstva, kao sastavni i bitan deo funkcionisanja servisa i usluga klaud okruženja kao celine, može da se sagleda kao skup hijerarhijski organizovanih komponenti, koje omogućavaju isporuku računarskih resursa u vidu usluge. Sistem klaud računarstva, koji obuhvata i implementacione detalje (tehnologiju) sastoji se iz četiri sloja (nivoa) [23]:

- prvi sloj - sloj korisnika/klijenta;
- drugi sloj - mrežni sloj;
- treći sloj - klaud menadžment i
- četvrti sloj - sloj hardverskih resursa.

Prvi sloj arhitekture klaud računarstva, sloj korisnika/klijenta, je najniži sloj, i kako sledi iz samog naziva, predstavlja sloj sa kog krajnji korisnici iniciraju konekciju sa klaud servisima. Ovaj sloj obuhata heterogene klijentske uređaje, koji se u opštem smislu sastoje od tankih i debelih klijenata. Tanki klijenti služe samo kao interfejs ka klaud servisima, dok debeli klijenti imaju mogućnost i obrade podataka u lokalu.

Mrežni sloj omogućava korisnicima (klijentima) da se povežu na klaud okruženje. Drugim rečima, mrežni sloj obuhvata mrežnu infrastrukturu i komponente koje povezuju klaud klijenta i klaud provajdera. Efikasnost isporučivanja klaud servisa značajno zavisi od pouzdanosti i propusne moći mrežne infrastrukture, kao komunikacionog medijuma između klaud provajdera i krajnjeg korisnika.

Način na koji će klijenti i provajderi biti povezani zavisi u velikoj meri od modela klaud računarstva koji se koristi. U scenarijima gde se koriste javne klaud usluge, osnovni medijum je javni Internet. Tačna lokacija javnog klauda se apstrahuje i klijenti ne znaju gde se fizički nalaze usluge kojima pristupaju. Međutim, dokle god klijenti dobijaju pravovremene i kvalitetne usluge i dokle god im je omogućen pristup sa bilo kojeg uređaja i sa bilo koje tačke u svetu, informacija o preciznoj lokaciji servisa nije relevantna.

U scenarijima kada se koristi privatni klaud, pristup klaudu unutar organizacije omogućen je lokalnom računarskom mrežom (eng. local area network - LAN), dok zaposleni i saradnici sa udaljene lokacije pristupaju privatnom klaudu najčešće korišćenjem tehnologija poput virtuelne privatne mreže (eng. virtual private network - VPN). U poslednje vreme, korišćenjem bezbednosnih protokola, poput SSTP (secure socket tunneling protocol), prenos podataka putem virtuelnog privatnog tunela kroz javni Internet je relativno bezbedan.

Koji god tip klaud servisa da se koristi (privatni, javni ili neki od druga dva modela), kada se govori o mrežnom sloju, najvažniju ulogu imaju provajderi klaud usluga, kojima je imperativ zadovoljavanje potreba krajnjih korisnika, a da pritom ne dođe do narušavanja klauzula definisanih u Ugovoru o korišćenju klaud usluga (eng. service level agreement - SLA).

Treći sloj klaud arhitekture, klaud menadžment sloj, obuhvata skup softvera i alata koji se koriste za održavanje klaud servisa i resursa, kao i za obezbeđivanje servisa i usluga krajnjim korisnicima. Sloja klaud menadžmenta omogućava klijentima da kroz

jednostavan (eng. user friednly) Web bazirani ili konzolni interfejs (eng. command line interface - CLI) pristupaju i konfigurišu svoje klaud servise, uz minimalnu, ili bez interakcije klaud provajdera. Funkcionalnosti ovakvih alata na najbolji način omogućavaju primene klauzula definisanih u SLA-u.

Poslednji, najviši sloj klaud arhitekture je sloj hardverskih resursa. Ovaj sloj omogućava i obezbeđuje hardverske resurse koji su neophodni za funkcionisanje klaud okruženja. U najvećem broju slučajeva u pitanju je hiper-konvergirana infrastruktura sa sofisticiranim hipervizorom i vrhunskim mrežnim i komponentama za skladištenje podataka. Uz simultano ostvarivanje ciljeva zadovoljstva korisnika (pre svega kvalitet i pravovremenost usluga) i racionalne upotrebe raspoloživih računarskih resursa, sloj hardverskih resursa je sloj koji pokreće celu klaud „mašineriju".

3 Pojam i kategorizacija optimizacionih problema

Prema jednoj definiciji optimizacije koja se često navodi u savremenoj literaturi, optimizacija je proces kojim se pronalazi maksimum ili minimum neke funkcije [85]. Optimizacija je našla široku primenu u svakodnevnom životu, zato što je za mnoge probleme i izazove iz realnog sveta potrebno da se nađu što bolja rešenja. Kao neki od primera navode se optimizacija saobraćaja, distribucije električne energije, rasporeda obavljanja poslova, lokacije senzora u bežičnim senzorskim mrežama, itd. U ovom radu optimizacija se vezuje za domene koji obuhvataju računarstvo, operaciona istraživanja i matematiku.

Naučna grana koja se bavi razvojem modela i metoda za rešavanje matematički formulisanih problema optimizacije naziva se teorija optimizacije (eng. optimization theory). Generalizacija teorije optimizacije i tehnike spada u širu oblast koja se naziva primenjena matematika (eng. applied mathematics).

Glavni cilj postupka optimizacije je pronalaženje optimalnog ili suboptimalnog (približno optimalnog) rešenja u odnosu na definisane ciljeve i potencijalna ograničenja. Prema nekim literaturnim izvorima, proces optimizacije može da se podeli na sledeće faze, koje se izvršavanju sekvencijalno [98]: definisanje problema, formulisanje i rešavanje matematičkog modela problema, implementacija i validacija rešenja.

Funkcija cilja (eng. objective function) predstavlja kvantitativnu meru performansi optimizacionih problema i kako bi proces optimizacije mogao da počne, neophodno je identifikovati i formulisati ovu funkciju. Takođe, funkcija cilja može da se definiše i kao mera kojom se utvrđuje kvalitet rešenja optimizacionog problema i koja omogućava poređenje različitih rešenja sa ciljem izbora najboljeg. Drugim rečima, funkcija cilja omogućava da se svakom vektoru (rešenju) pridruži odgovarajuća vrednost koja predstavlja njegovu meru kvaliteta.

Formulacija funkcije cilja zavisi od optimizacionog problema koji je potrebno rešiti i ona može da bude na primer vrednost neke funkcije, ostvareni profit, najmanji broj elemenata u nekom sistemu, ili bilo koja druga mera ili njihova kombinacija koja se prikazuje kao jedan broj [85].

Karakteristike optimizacionog problema nazivaju se promenljive ili varijable. Potrebno je pronaći vrednosti promenljivih koje optimizuju formulisanu funkciju cilja. Za veliki broj optimizacionih problema iz svakodnevnog života varijable mogu da

imaju samo određene vrednosti, dok s druge strane značajan broj problema uključuje i ograničenja.

Termin koji je u tesnoj vezi sa optimizacijom je modeliranje, koji se definiše kao proces kojim se utvrđuje funkcija cilja, varijable i potencijalna ograničenja postavljenog optimizacionog problema [85]. U formulisanju modela, posebno praktičnih problema, treba da se vodi računa o njegovoj složenosti. U prvom slučaju, ukoliko je formulisani model previše jednostavan, on neće moći realno da modelira sva svojstva problema na koji se odnosi. S druge strane, u slučaju previše kompleksnog modela, ni jedna optimizaciona metoda neće moći uspešno da ga reši. Često se u formulacijama modela problema iz realnog okruženja koriste aproksimacije, ali na način da se zadrže bitna svojstva datog problema. Odgovarajuća optimizaciona metoda primenjuje se nakon što se model formuliše.

U procesu rešavanja optimizacionog problema, vrši se izbor rešenja (alternative) koja maksimizuje ili minimizuje funkciju cilja i zadovoljava sva ograničenja. Kada se govori o optimizaciji, u literaturi se obično razmatra minimizacija, što je primenjeno i u ovoj disertaciji.

Problem optimizacije za specijalni slučaj R^n može da se matematički formuliše na sledeći način:

$$\begin{aligned} &\text{za dato } f : R^n \to R \\ &\text{pronaći } x^* \in R^n, \text{ uz zadovoljenje uslova,} \\ &\forall x \in R^n,\ f(x^*) \leq f(x) \end{aligned} \tag{1}$$

gde R^n označava prostor varijabli (prostor pretrage) funkcije f, notacija $x \in R^n$ označava potencijalno rešenje problema, dok je optimalno rešenje označeno sa x^*. Promenljiva n se odnosi na veličinu prostora pretrage u smislu broja dimenzija optimizacionog problema i u ovoj formulaciji predstavlja broj varijabli funkcije cilje.

3.1 Taksonomija optimizacionih problema

Za potrebe taksonomije problema optimizacije mogu se koristiti različiti kriterijumi i shodno tome je moguće napraviti veći broj klasifikacija. U nastavku su prikazane

samo one kategorizacije koje su relevantne za istraživanje koje je sprovedeno za potrebe ove doktorske disertacije.

Razmatrana je klasifikacija optimizacionih problema na osnovu tri kriterijuma: prema tipu promenljivih, u odnosu na kriterijum da li optimizacioni problem ima ili nema ograničenja i na osnovu broja funkcija cilja.

3.1.1 Kategorizacija prema tipu varijabli

Kada se uzima u obzir kriterijum tip varijabli, optimizacioni problemi se dele na kombinatorne (eng. combinatorial optimization) i na kontinualne (eng. continuous optimization), gde u prvom slučaju promenljive uzimaju diskretne, a u drugom neprekidne (realne) vrednosti.

U slučaju problema diskretne (kombinatorne) optimizacije, postoji ograničenje da promenljive mogu da uzmu samo određene vrednosti iz skupa realnih brojeva. Obično se radi o skupu prirodnih brojeva, skupu prirodnih brojeva sa nulom ili binarnom skupu.

Opšta formulacija diskretne optimizacije [62]: zadat je konačan skup $E = e_1, e_2, ..., e_n$. Za svaki element skupa $e \in E$ definisana je funkcija $\omega(e)$ takva da [62]:

$$\omega : E \to R \tag{2}$$

Dopustivi skup problema kombinatorne optimizacije se definiše kao [62]:

$$X \subseteq P(E) \tag{3}$$

Pri čemu je svako dopustivo rešenje $x \in X$ zapravo $x \subseteq E$. Funkcija cilja problema diskretne optimizacije najčešće se formuliše kao [62]:

$$f(x) = \sum_{e \in x} \omega(e), \tag{4}$$

dok se optimizacioni problem diskretne optimizacije formuliše kao [62]:

$$\begin{aligned} &\min \sum_{e \in x} \omega(e) \\ &p.o. \\ &x \in X \end{aligned} \tag{5}$$

Na osnovu navedenog, optimizacioni problem diskretne optimizacije u opštem slučaju može da se formuliše kao [62]:

$$\begin{aligned} &\min(\max) f(x) \\ &p.o. \\ &x \in X, \end{aligned} \tag{6}$$

gde je X konačan (prebrojiv) skup i važi $f(x) : X \to R$.

Veliki broj praktičnih problema spada u kategoriju kombinatorne optimizacije, kao na primer transporni problemi, problem izbora projekata i portfolija finansijske aktive, raspoređivanja stvari u ranac i poslova na raspoložive virtuelne mašine na klaudu, itd.

Kontinualna (neprekidna) optimizacija obuhvata sve probleme kod kojih promenljive mogu da imaju realne vrednosti. Kao poseban slučaj kontinualne optimizacije, gde postoji jako veliki broj lokalnih optimuma, koje nije moguće sve pretražiti i gde je potrebno da se pronađe globalno najbolje rešenje (eng. global best solution) navodi se globalna optimizacija (eng. global optimization). Ovaj tip optimizacije ukazuje na to da može da se definiše lokalni optimum, ali on važi na samo ograničenom intervalu posmatranja. Ukoliko se posmatra ceo interval može da postoji više lokalnih optimuma i jedan ili više globalnih optimuma.

Za rešavanje problema globalne optimizacije najčešće se primenjuju metaheuristike, koje izvršavaju proces pretrage pomoću procesa intenzifikacije i diversifikacije, i za veliki broj praktičnih problema uspevaju da pronađu optimalno ili zadovoljavajuće rešenje.

Takođe je korisno da se napomene da neke metode optimizacije, kao što su metode lokalne pretrage (eng. local search methods) traže samo tačku čija je funkcija cilja manja od funkcije cilja tačaka koje se nalaze u njenoj okolini (susedstvu) [85]. Poznatiji primeri takvih metoda, koje ne mogu da se primene na probleme globalne

optimizacije, zato što ne pronalaze globalni, već lokalni optimum, su metoda penjanja uz brdo (eng. hill-climbing method) i Njutnov metod (eng. Newton method).

Opšta formulacija problema globalne kontinualne optimizacije data je kao:

$$\min \ f(x), \tag{7}$$

gde $x \in R^n$ predstavlja realni vektor sa $n \geq 1$ komponentom, dok funkcija koja se minimizuje ima oblik: $f : R^n \to R$. U slučaju ovakvih problema, optimizacioni metod nema znanje o funkciji cilja f, već samo ima mogućnost da izračuna njenu vrednost i proizvoljnoj tački.

3.1.2 Podela na osnovu ograničenja

Optimizacioni problemi takođe se mogu klasifikovati i u zavisnosti od toga da li postoje ograničenja, gde se razlikuju problemi uslovne optimizacije (optimizacija sa ograničenjima) i bezuslovne optimizacije (optimizacija bez ograničenja).

Kod klase problema bezuslovne optimizacije (eng. unconstrained optimization), promenljive uzimaju vrednosti u okviru dozvoljenih donjih i gornjih granica, zbog čega se ovaj tip problema u literturi još naziva i bezuslovna optimizacija sa ograničenjima vrednosti promenljivih (eng. bound-constrained optimization).

Problemi kontinualne bezuslovne optimizacije (eng. continious global unconstrained optimization) mogu da se formulišu na sledeći način:

$$\min \ f(x), \ x = (x_1, x_2, ..., x_n) \in S \subseteq R^n, \tag{8}$$

gde $S \subseteq R^n$ predstavlja prostor pretrage, dok je n broj dimenzija problema. Dalje, S je n-dimenzioni hiper-kvadar u prostoru R^n koji je definisan pomoću donjih i gornjih granica vrednosti promenljivih:

$$lb_i \leq x_i \leq ub_i, \ 1 \leq i \leq n, \tag{9}$$

gde su ub_i i lb_i gornja i donja granica parametra i, respektivno.

Tačka $x^* \in R^n$ koja zadovoljava ograničenja vrednosti parametara (jednačina (9)) i u slučaju problema minimizacije uslov da je:

$$f(x^*) \leq f(x),\ \forall x \in R^n \tag{10}$$

naziva se optimalno rešenje problema kontinulane bezuslovne optimizacije.

Ukoliko se za rešavanje optimizacionog problema postave ograničenja, dopustivi region prostora pretrage se znatno sužava. Rešenja koja zadovoljavaju ograničenja nazivaju se dopustiva (eng. feasible), dok se rešenja koja ne zadovoljavaju ograničenja nazivaju nedopustiva (eng. infeasible).

Kontinualna uslovna optimizacija (eng. continuous constrained optimization) definiše se na sledeći način:

$$\min \text{ (ili } \max) \ f(x),\ x = (x_1, x_2, ..., x_n), \tag{11}$$

gde je $x \in F \subseteq S \subseteq R^n$, dok S predstavlja hiper-kvadar određen jednačinom (9).

Domen prostora pretrage sa dopustivim rešenjima $F \subseteq S$ određuje se pomoću skupa od m linearnih i nelinearnih ograničenja:

$$\begin{aligned} g_j(x) \leq 0,\ \text{za } j = 1, ..., q \\ h_j(x) = 0,\ \text{za } j = q+1, ..., m, \end{aligned} \tag{12}$$

gde q označava ukupan broj ograničenja nejednakosti (eng. inequality constraints), dok $(m-q)$ predstavlja ukupan broj ograničenja sa jednakostima (eng. equality constraints). Osim navedenog, u svakoj tački $x \in F$ sva ograničenja g_k koja zadovoljavaju jednakost $g_k(x) = 0$ nazivaju se aktivna ograničenja u tački x.

Tačka $x^* \in S$ koja zadovoljava sva ograničenja (jednačina (12))i u slučaju problema minimizacije uslov da je:

$$f(x^*) \leq f(x),\ \forall x \in S \tag{13}$$

naziva se globalno optimalno rešenje problema.

Da bi mogli uspešno da rešavaju probleme sa ograničenjima, optimizacioni algoritmi najčešće inkorporiraju neku od metoda za upravljanje ograničenjima. Kao neke od najčešće korišćenih kategorija tehnika za upravljanje ograničenjima navode se

sledeće [79], [33], [28], [78]:

- kaznene funkcije;
- Debova pravila;
- posebni operator;
- odvajanje funkcije cilja i ograničenja i
- dekoderi.

3.1.3 Kategorizacija prema broju funkcija cilja

Ako se u uzme u obzir broj funkcija funkcija cilja (kriterijuma), problemi optimizacije mogu da se podele na probleme sa jednom funkcijom cilja (eng. single-objective optimization) i na probleme sa više funkcija cilja (eng. multi-objective optimization - MOO). Optimizacija sa jednom funkcijom cilja poznata je u literaturi i kao jednokriterijumska optimizacija, dok se za optimizaciju sa više funkcija cilja koriste termini višekririjumska optimizacija i više-ciljno programiranje.

U slučaju problema sa više funkcija cilja, postoji više ciljeva koje je potrebno simultano optimizovati, gde su ciljevi najčešće kontradiktorni (u suprotnosti) jedni drugima. Na primer, sa poboljšanjem jednog cilja, drugi cilj se pogoršava i obrnuto.

Problemi višekriterijumske optimizacije se često sreću u svakodnevnom životu u raznim oblastima, poput menadžmenta, finansija i primenjenog računarstva. Tako na primer, u slučaju optimizacije finansijskog portfolija potrebno je uvećati prinos na finansijsku aktivu uz minimizaciju rizika. U domenu bežičnih senzorskih mreža potrebno je simultano ostvariti najveću pokrivenost senzorima uz minimizaciju potrošnje električne energije.

Postoji više načina za rešavanje višekriterijumskih problema, među kojima su sledeći: pristup težinske sume (eng. weighted sum) [72], hijerarhija ciljeva [69] i koncept Pareto optimalnosti [22]. Zbog relevantnosti sa istraživanjem koje je prikazano u doktorskoj disertaciji, u nastavku su ukratko opisana prva dva navedena pristupa.

Primenom pristupa težinske sume, svi ciljevi se grupišu (agregiraju) u jedan, množenjem vrednosti svakog cilja predefinisanim težinskim koeficijentima. Glavno pitanje koje se postavlja prilikom primene ovog metoda jeste koji težinski koeficijent

dodeliti svakom cilju. Ova odluka zavisi od prioriteta i značaja određenog cilja. Takođe, u mnogim praktičnim problemima funkcije cilja imaju numeričke vrednosti iz različitih opsega i u tom slučaju potrebno je izvršiti skaliranje.

Pristup težinske sume matematički može da se formuliše na sledeći način: skalarni težinski koeficijent ω bira se za svaki cilj $f_1(x), f_2(x), f_3(x)..., f_n(x)$ i rešavanje problema svodi se na optimizaciju složene funkcije cilja C [72]:

$$C = \sum_{i=1}^{n} \omega_i f_i(x) \tag{14}$$

Korišćenjem tehnike hijerarhije ciljeva, svakom cilju se dodeljuje prioritet, gde definisani prioritet zavisi od problema koji se rešava. Zatim se prvo optimizuje cilj sa najvećim prioritetom (najvažniji cilj) i tek nakon toga se sprovodi proces optimziacije ciljeva sa drugim, trećim, četvrtim, itd. prioritetom. Ovaj pristup je našao široku primenu u mnogim praktičnim problema iz domena elektrotehnike i računarstva, poput problema planiranja mreže identifikacije pomoću radio frekvencija (eng. radio frequency identification (RFID) network planning problem - RNP) [69].

Više o konceptu Pareto optimalnosti može se naći u [22].

3.2 Algoritmi i klase kompleksnosti

Pre uvođenja klasa kompleksnosti, potrebno je osvrnuti se na složenost algoritama, posebno na vremensku i prostornu komponentu složenosti.

U savremenoj literturi uveden je koncept kvantifikacije vremena koje potrebno da se algoritam izvrši zbog toga što se neretko dešava da jedan isti algoritam u razumnom vremenskom periodu na manjim instancama problema (manji broj ulaza) uspe da generiše rezultat, dok na većim instancama problema (veći broj ulaza) njegovo izvršavanje traje neprihvatljivo dugo i zbog toga je praktično neupotrebljiv.

Vreme izvršavanja algoritma poznato je kao vremenska složenost, koje se računa kao broj koraka izračunavanja (eng. computation steps) koje je potrebno algoritam da izvrši, kako bi završio određeni zadatak. Međutim, obzirom da se pod korakom izračunavanja podrazumevaju različite operacije, poput množenja i deljenja, koje traju različito, kao i da vreme koje je potrebno da se koraci izvrše zavisi i od računarskog sistema na kojem se algoritam izvršava, za svaki algoritam se određuje osnovni korak,

čije je trajanje jedna vremenska jedinica. Zatim se u odnosu na osnovni definisani korak kasnije vrši evaluacija vremenske složenosti celog algoritma.

Osim vremena izvršavanja, važna je i prostorna složenost algoritma, koja se odnosi na potrošnju memorijskih resursa, tj. na veličinu memorije koja je neophodna da bi se algoritam izvršio. Prilikom evaluacije prostorne složenosti, skladišni kapacitet koji je potreban za čuvanje ulaznih podataka se u većini slučajeva zanemaruje.

Skup individualnih zadataka čini optimizacioni problem, gde individualni zadatak predstavlja instancu problema koji se optimizuje sa konkretnim parametrima. Kaže se da se problem možete rešiti algoritmim (algoritamski rešiv problem) ako i samo ako je raspoloživ algoritam koji uspešno može da se primeni na svaki pojedinačni zadatak optimizacionog problema i pritom takav algoritam treba da reši zadatak u razumnom vremenskom periodu korišćenjem raspoloživih računarskih resursa.

Taksonomija optimizacionih problema na osnovu kriterijuma težine problema definisana je u teoriji kompleksnosti izračunavanja (eng. computation complexity theory). Minamalni obim (količina) prostornih i vremenskih računarskih resursa koji su neophodni za rešavanje problema čine osnovu za definisanje težine problema [7], gde je težina problema u uskoj vezi sa složenošću algoritma. Zbog toga, polazeći od vremenske i prostorne složenosti algoritma, moguće je odrediti gornju i donju granicu težine problema [98].

Klasi složenosti P (eng. polynomial time) pripadaju oni problemi koji u opštem slučaju mogu biti rešeni primenom nekih od dostupnih algoritama polinomske složenosti. S druge strane, NP (eng. non-deterministic polynomial time) klasa kompleksnosti odnosi se na grupu problema čija se rešenja mogu proveriti (verifikovati) primenom algoritma polinomske složenosti.

Za istraživanje koje je prikazano u ovoj disertaciji posebno je značajna klasa NP teških (eng. NP hard) problema, koja obuhvata probleme za koje ne postoji polinomski algoritam pomoću kojeg se mogu rešiti i koji mogu da se transformišu jedan u drugi u polinomskom vremenu. Drugačije rečeno, algoritam koji se koristi za rešavanje NP teškog problema može se u polinomskom vremenu redukovati na drugi algoritam koji može da reši bilo koji problem iz klase NP [98].

Takođe, potrebno je da se spomenu i NP kompletni (eng. NP complete) problemi. Problem spada u klasu NP kompletnih problema ako pripada klasi NP i ako algoritmom koji ima polinomsku složenost svi ostali NP problemi mogu da se svedu na njega.

4 Modeli raspoređivanja poslova i balansiranja opterećenja na klaudu

Kao što je već i Poglavlju 1, za potrebe istraživanja u ovoj doktorskoj disertaciji, u simulacijama su korišćeni veštački (generisani u okviru okruženja CloudSim simulatora) i realni podaci, koji su preuzeti iz svetskih benčmark baza podataka iz ove oblasti. Osim toga, korišćena su i dva modela raspoređivanja poslova i balansiranja opterećenja u klaud okruženju. Oba modela spadaju u grupu kombinovanih NP teških problema kombinatorne i globalne optimizacije.

Prvi model spada u grupu optimizacionih problema sa jednom funkcijom cilja, gde je uzeta u obzir minimizacija ukupnog vremena izvršavanja svih poslova (eng. makespan - MS) na raspoloživim klaud računarskim resursima. Rešavanje istog modela primenom heurističkih i metaheurističkih metoda prikazano je u [34].

Drugi model raspoređivanja poslova i balansiranja opterećenja, koji je rešavan za potrebe ove doktorske disertacije spada u grupu višekriterijumske optimizacije sa ograničenjima, gde su uzeti u obzir dve funkcije cilja: ukupno vremene izvršavanja svih poslova i minimizacija troškova u okviru raspoloživog budžeta (eng. budget cost). Kao i u slučaju prvog modela, rešavanje drugog modela implementacijom i adaptiranjem heurističkih i metaheurističkih algoritama prikazano je u [102], [92].

Napominje se da, s obzirom da je u svetskoj literaturi ukupno vreme izvršavanja svih poslova poznato pod engleskim terminom makespan, je u nastavku teze ovaj termin uniformno korišćen.

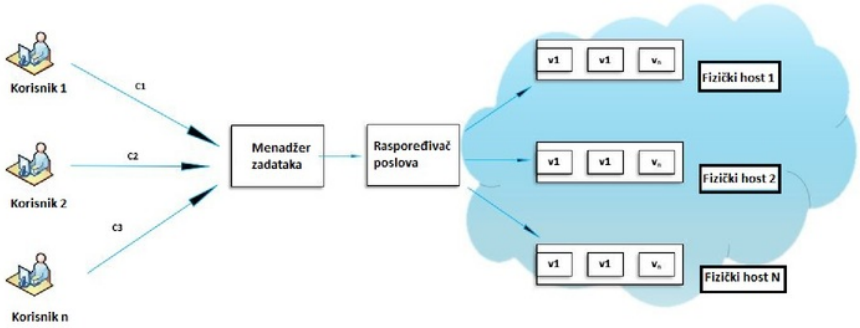
U nastavku su prikazane matematičke formulacije oba navedena modela zajedno sa osnovnom terminologijom i principima raspoređivanja poslova u klaud okruženju.

4.1 Raspoređivanje poslova u klaud okruženju i algoritmi

Pojam raspoređivanja poslova je važan aspekt klaud računarstva, gde je glavni cilj da se izvrše sve poslati zadaci u što kraćem vremenskom periodu i uz što manje korišćenja raspoloživih računarskih resursa. Preciznije, cilj je da se odredi najbolji resurs (pod resursom se uglavnom podrazumeva virtuelna mašina) za izvršavanja svakog zahteva koji je poslat od strane kranjeg korisnika. Navedeni ciljevi se ostvaruju primenom algoritma za raspoređivanje poslova. Na taj način postižu se poboljšanja

u mnogim aspektima kvaliteta usluge (eng. quality of service - QoS), kao što su pouzdanost, produktivnost, korišćenje resursa, potrošnja električne energije, troškovi izvršavanja, itd [82].

Sistemski model raspoređivanja poslova (zadataka) u klaud okruženju prikazan je na Slici 8.



Slika 8: *Model sistema raspoređivanja zadataka u klaud okruženju*

Primenom algoritama za raspoređivanje poslova, optimalna ili približno optimalna alokacija (mapiranje) raspoloživih resursa na poslate zadatke može da se ostvari u ograničenom vremenskom intervalu, kako bi se ostvario određeni nivo kvaliteta usluge [48]. Algoritmi kreiraju plan, u kome se određuje na kojim resursima i kada će svaki zadatak biti izvršen.

Kada se govori o raspoređivanju zadataka, pojam obezbeđivanja resursa (eng. resource provisioning) takođe treba da se pomene, zato što oba pojma pripadaju skupu aktivnosti upravljanja resursima (eng. resource management) u klaud okruženju.

Obezbeđivanje resursa je koncept koji se odnosi na određivanje i distribuciju virtuelizovanih računarskih resursa kranjim korisnicima, uz uspostavljanje optimalnog broja virtuelnih mašina sa odgovarajućim konfiguracijama i karakteristikama, za alokaciju po zahtevu [64]. Broj obezbeđenih (generisanih) virtuelnih mašina treba da bude na najmanjem, ali zadovoljavajućem nivou, kako bi kvalitet usluga kranjih korisnika ne bi bio narušen [44]. U skladu s navedenim, proces obezbeđivanja resursa se dalje deli na detekciju i izbor odgovarajućih resursa.

Kao što je već i navedeno, kada se vrši raspoređivanje resursa može doći do problema. Efikasni i efektivni algoritmi za obezbeđivanje resursa mogu da „pomognu”

algoritmu raspoređivanja, tako što analiziraju i organizuju opterećenje klaud sistema, koje zavisi od tipa zahteva kranjih korisnika, na optimalan način. U klasičnom modelu klaud računarstva plaćanja na osnovu nivoa ostvarene potrošnje, gde kranji korisnici šalju heterogene zahteve i kada je cilj klaud provajdera da obezbedi pristup resursima u zadovoljavajućem vremenskom periodu, raspoređivanje na zahtev (eng. on-demand scheduling) daje dobre rezultate. Raspoređivanje na zahtev je u savremenoj literaturi poznato još kao i online raspoređivanje (eng. online scheduling).

Međutim, primena online raspoređivanja može da dovede do problema. U scenarijima kada je distribucija opterećenja (koje zavisi od broja i tipa zahteva kranjih korisnika) na virtuelne mašine nejednaka, dolazi do situacije kada potrebe zahteva prevazilaze kapacitete određenih virtuelne mašine, što na kraju dovodi do prekoračenja vremena za izvršavanje zadataka. U literaturi je ovo poznato kao pitanje balansiranja opterećenja i zbog toga je jako važno da primenjeni algoritam za raspoređivanje poslova uzme u obzir i ravnopravnu distribuciju opterećenja na raspoložive virtuelne mašine.

S druge strane, u scenarijima gde se koristi veći ili manji broj virtuelnih mašina od realnih potreba za resursima, takođe dolazi do problema. Prvi slučaj, kada je broj raspoloživih virtuelnih mašina manji od realnih zahteva poslatih zadataka u literaturi je poznat kao scenario nedovoljnog obezbeđivanja (eng. under provisioning), dok je drugi slučaj, kada je broj raspoloživih virtuelnih mašina većih od potreba zahteva kranjih korisnika poznat je pod nazivom prekomerno obezbeđivanje (eng. over provisioning). U prvom slučaju kvalitet usluga opada, što dovodi do nezadovoljstva krajnjih korisnika, dok se u drugom slučaju javlja efekat da dolazi bespotrebnog rasta troškova korišćenih resursa.

Prema jednoj taksonomiji, algoritmi za raspoređivanje poslova se dele na statičke i dinamičke [82] [64]. Pre samog izvršavanja statičkog algoritma, potrebno je prikupiti detaljne informacije koje se odnose na zadatke koje je potrebno izvršiti, kao i na resurse (virtuelne mašine) koje je potrebno obezbediti za izvršavanje poslatih zadataka. Informacije koje se odnose na zadatke najčešće obuhvataju sledeće: broj zadataka, dužina zadataka, zahtevi za resursima izračunavanja, skladištenja i mrežnim resursima, rokovi za izvršavanje, itd. S druge strane, osnovni podaci o virtuelnim mašinama uključuju raspoložive kapacitete jedinica za izračunavanje, memorije, skladišta i

mrežnog propusnog opsega, potrošnju električne energije, itd. Zbog dinamičke i heterogene prirode klaud okruženja, kao i zbog različitosti zahteva kranjih korisnika, statički algoritmi ne daju zadovoljavajuće rezultate kada se primenjuju u sistemima kao što je klaud, zbog toga što nisu u stanju da optimalno (ili približno optimalno) rasporede opterećenja na raspoložive resurse.

Osim navedenog, tokom praktične eksploatacije statičkih algoritama, zaključeno je da ovi algoritmi negativno utiču na osnovne indikatore kvaliteta usluga (makepsan, pouzdanost i dostupnost). Najčešće korišćeni statički algoritmi raspoređivanja su svako sa svakim (eng. round robin - RR), prvi-u-prvi-van (eng. first in first out - FIFO) i najkraći posao prvi (eng. shortest job first - SJF).

Zbog navedenih nedostataka statičkih algoritama raspoređivanja, za ove potrebe u klaud okruženju je bolje implementirati dinamičke algoritme raspoređivanja (eng. dynamic scheduling algorithms). Za razliku od statičkih algoritama, koji se baziraju na podacima o zahtevima i resursima, dinamički algoritmi se zasnivaju na praćenju i na dinamičkom usklađivanju izvršavanja zadataka u realnom vremenu. Ovi algoritmi neprestano prate promene u klaud okruženju i usklađuju opterećenje između različitih čvorova (virutlenih mašina) u skladu sa uslovom preopterećenja (eng. overloaded condition).

Dinamički algoritmi raspoređivanja su u mogućnosti da uspostave dobro usklađivanje opterećenja u klaud sistemu uz manji novi disbalansa između virtuelnih mašina (čvorova). Kao neki od primera dinamičkih algoritama raspoređivanja navode se heterogeno najranije vreme završetka (eng. heterogeneous earliest finish time - HEFT), heterogeno grupisanje sa umnožavanjem (eng. clustering based heterogeneous with duplication - CBHD) i algoritam najmanje ponderisane veze (eng. weighted least connection - WLC). Takođe, kao što je već i navedeno u poglavlju 1, metaheuristički algoritmi, posebno metaheuristike inteligencije rojeva, su se pokazali kao efikasni dinamički pristupi za raspoređivanje poslova na klaudu. Pregled literature primene algoritama inteligencije rojeva iz domena raspoređivanja poslova na klaudu i drugih NP teških problema, dat je u Poglavlju 6.3.

Sa neprestanim rastom broja i kompleksnosti zadataka koje je potrebno izvršiti u klaud okruženju, raste i potreba za sofisticiranim algoritmima za raspoređivanje poslova. U ovom domenu postoji veliki potencijal, kako za unapređenja postojećih,

tako i za kreiranje i implementiranje novih algoritama. Efikasno raspoređivanje poslova treba da bude jedan od glavnih prioriteta na klaudu, zato što je imperativ zadovoljavanje potreba krajnjih korisnika, a da pritom ne dođe do narušavanja klauzula definisanih u SLA Ugovoru.

4.2 Model raspoređivanja resursa u klaud okruženju sa jednom funkcijom cilja

Prvi model raspoređivanja resursa na klaudu, koji je rešavan za potrebe istraživanja koje je prikazano u ovoj doktorskoj disertaciji spada u grupu modela sa jednom funkcijom cilja. Sličan model korišćen je i u istraživanju koje je prikazano u radu [34], koji je objavljen u vrhunskom međunarodnom časopisu kategorije M21a.

Da bi se prikazala matematička formulacija modela raspoređivanja sa jednom funkcijom cilja uvedene su sledeće notacije. Neka CI označava klaud infrastrukturu koja se sa sastoji iz N_{ph} fizičkih računara (PH), gde se na svakom hostu (fizičkom računaru) izvršava N_{vm} virtuelnih mašina (VM):

$$CI = \{PH_1, PH_2, ..., PH_i, ..., PH_{N_{ph}}\}, \tag{15}$$

where each $PH_i (i = 1, 2, 3, ..., N_{ph})$ can be denoted as:

$$PH_i = \{VM_1, VM_2, ..., VM_j, ..., VM_{N_{vm}}\}, \tag{16}$$

gde VM_j označava j-tu virtuelnu mašinu koja se alocira u okviru određenog host računara. Performanse svake VM_j virtuelne mašine se dalje preciznije određuju sledećim karakteristikama (atributima, svojstvima):

$$VM_j = \{VMID_j, MIPS_j\}, \tag{17}$$

gde $VMID_j$ i $MIPS_j$ označavaju jedinstveni identifikator (eng. unique identifier), tj. serijski broj (eng. serial number) i performanse obrade obrada u MIPS jedinicama virtuelne mašine VM_j, respektivno.

Krajnji korisnici šalju skup zadataka (TSK) za izvršavanje na klaudu, koji se dalje dodeluju pomoću algoritma raspoređivanja raspoloživim i odgovarajućim virtuelnim

mašinama:

$$TSK = \{T_1, T_2, T_3, ..., T_k, ..., T_{N_{tsk}}\}, \tag{18}$$

gde N_{tsk} označava ukupan broj zadataka koji su poslati na klaud, dok notacija T_k predstavlja k-ti zadatak, koji se dalje preciznije određuje kao:

$$T_k = \{TID_k, length_k, ETC_k, P_k\}, \tag{19}$$

gde TID_k i $length_k$ označavaju jedinstveni identifikator i dužinu zadatka k, koja se izražava u jedinici milioni instrukcija (eng. million instructions - MI).

Alokacija (mapiranje) skupa od TSK zadataka na N_{vm} virtuelnih mašina ima veliki uticaj na ukupne performanse klaud sistema.

Očekivano vreme završetka (eng. expected time to complete) zadatka k na j-toj virtuelnoj mašini $ETC_{k,j}$ može da se izračuna na sledeći način:

$$ETC_{k,j} = \frac{length_k}{MIPS_j}, \tag{20}$$

where $j = 1, ..., N_{vm}$ and $k = 1, ..., N_{tsk}$.

Matricom ETC očekivanog vremena završetka svih zadataka veličine $N_{tsk}xN_{vm}$ definiše se koiliko je vremena potrebeno kako bi se svaki zadatak izvršio na svakoj virtuelnoj mašini koja je raspoloživa u klaud sistemu [34]:

$$ETC = \begin{bmatrix} ETC_{1,1} & ETC_{1,2} & ETC_{1,3} & \cdots & ETC_{1,j} & \cdots & ETC_{1,N_{vm}} \\ ETC_{2,1} & ETC_{2,2} & ETC_{2,3} & \cdots & ETC_{2,j} & \cdots & ETC_{2,N_{vm}} \\ ETC_{3,1} & ETC_{3,2} & ETC_{3,3} & \cdots & ETC_{3,j} & \cdots & ETC_{3,N_{vm}} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ ETC_{N_{tsk},1} & ETC_{N_{tsk},2} & ETC_{N_{tsk},3} & \cdots & ETC_{N_{tsk},j} & \cdots & ETC_{N_{tsk},N_{vm}} \end{bmatrix}. \tag{21}$$

Ova matrica se široko koristi u literaturi koja se bavi raspoređivanjem poslova u klaud okruženju i predstavlja standardni način za modeliranje izvršavanje zadataka na klaud infrastrukturi sa heterogenim resursima [82].

Da bi se formulisala funkcija cilja minimizacije makespan indikatora, prvo je potrebno da izračuna koliko je potrebno da svaka virtuelna mašina obradi sve zadatke

koje su joj dodeljeni, tj. da se izračuna vreme izvršavanja (eng. execution time - ET) svake virtuelne mašine.

Vreme izvršavanja virtuelne mašine j za zadatak k zavisi od promenljive odluke (eng. decision variable) $x_{k,j}$ [82]:

$$x_{k,j} = \begin{cases} 1, \text{ if } C_k \text{ is allocated to } VM_j, \\ 0, \text{ if } C_k \text{ is not allocated to } VM_j. \end{cases} \tag{21}$$

Nakon toga, ET_j, gde je j u rasponu $[1, N_{vm}]$, može da se izračuna na sledeći način:

$$ET_j = \sum_{k=1}^{N_{tsk}} x_{k,j} \cdot ETC_{k,j}. \tag{22}$$

Konačno, funkcija cilja minimizacija makespan vrednosti (MS) predstavlja najveću (maksimalnu) vrednost ET svih virtuelnih mašina:

$$MS = \max[ET_j]_{j=1}^{N_{vm}}, \tag{23}$$

$$\forall k \in [1, N_{tsk}] \text{mapped to } \ j\text{th VM}, j = 1, 2, 3, ..., N_{vm}. \tag{24}$$

Osim navedenog, sa ciljem preciznije evaluacije performansi predloženih metaheuristika, uzet je u obzir i indikator stepena disbalanca (DI), koji se računa na sledeći način:

$$DI = \frac{T_{max} - T_{min}}{T_{avg}}, \tag{25}$$

gde T_{max}, T_{min} i T_{avg} označavaju najveće, najmanje i prosečno vreme izvršavanja svih virtuelnih mašina, respektivno. Ovaj indikator je takođe uzet u obzir u istraživanju koje je prikazano u radu [34].

Rezultati koji su ostvareni primenom metaheuristika rojeva za rešavanje ovog modela raspoređivanja poslova na klaudu publikovani su u [121] i [114].

4.3 Model raspoređivanja resursa u klaud okruženju sa više funkcija cilja

Drugi model raspoređivanja resursa na klaudu, koji je rešavan za potrebe istraživanja koje je prikazano u ovoj doktorskoj disertaciji, spada u grupu optimizacije sa ograničenjima sa više funkcija cilja. U ovom modelu, uzeta je u obzir minimizacija makespan indikatora, kao i minimizacija ukupnih troškova raspoloživog budžeta (eng. budget cost objective). Sličan model korišćen je i u simulacijama, čiji su rezultati prikazani u radovima [102] i [92], koji su objavljeni u vrhunskim međunarodnim časopisima.

Ovaj model je u literaturi iz domena računarstva poznat kao model raspoređivanja resursa baziran na performansama i ograničenjima raspoloživog budžeta (eng. model based on the performance and budget constraints) [92]. Performanse se određuju na osnovu makespan indikatora, uz zahteve da se svi zadaci završe pre unapred zadatog roka i da troškovi izvršavanja zadataka budu u okviru predefinisanih budžetskih ograničenja.

U ovom modelu uzimaju se u obzir troškovi izvršavanja virtuelnih mašina, koji se određuju troškovima procesorskog vremena (eng. central processing unit - CPU cost) i memorije (eng. memory costs). Ukupni troškovi budžeta izračunavaju se kombinacijom navedene dve kategorije troškova. Podobnost svakog rešenja izračunava se sabiranjem vrednosti ukupnog vremena izvršavanja svih zadataka (makespan) i ukupnih troškova budžeta.

Podobnost zavisi od kvaliteta rešenja. Svako rešenje treba da ima što manju vrednost makespan indikatora i funkcije koja modelira troškove budžeta. Prvo se računaju troškovi CPU vremena virtuelne mašine j pomoću sledećeg izraza:

$$C_{(x)} = \sum_{j=1}^{N_{VM}} C^{cost}(j), \tag{26}$$

gde $C^{cost}(j)$ označava troškove procesora virtuelne mašine VM_j, dok N_{VM} predstavlja ukupan broj virtuelnih mašina u klaud infrastrukturi. Notacija x označava izvodljivo (eng. feasible) rešenje. Vrednost $C^{cost}(j)$ se dalje izračunava na sledeći način:

$$C^{cost}(j) = C_{base} \times C_j \times t_{ij} + C_{Trans}, \tag{27}$$

gde C_{base} označava osnovu troškova, C_j predstavlja toškove CPU vremena virtuelne mašine VM_j, dok se t_{ij} odnosi na vreme tokom kojeg se zadatak T_i izvršava na resursu R_j (virtuelnoj mašini VM_j). Osim navednog, notacija C_{Trans} označava troškove prenost procesora (CPU). Osnovni troškovi C_{base} i C_{Trans} su konstantne vrednosti:

$$C_{base} = 0.17/hr, \tag{28}$$

$$C_{Trans} = 0.005, \tag{29}$$

Za izračunavanje funkcije troškova memorije, koristi se sledeći izraz:

$$M_{(x)} = \sum_{j=1}^{N_{VM}} M^{cost}(j), \tag{30}$$

gde $M^{cost}(j)$ označava trošak memorije virtuelne mašine VM_j i N_{VM} označava ukupan broj virtuelnih mašina. Dalje, $M^{cost}(j)$ se određuje kao:

$$M^{cost}(j) = M_{base} \times M_j \times t_{ij} + M_{Trans}, \tag{31}$$

gde se osnovni trošak memorije označava kao M_{base}, memorije virtuelne mašine VM_j se reprezentuje kao M_j , dok t_{ij} označava vreme koje je potrebno da bi se zadatak T_i obradio na resursu R_j (virtuelnoj mašini VM_j). Oznaka M_{Trans} predstavlja trošak prenosa memorije. Vrednosti M_{base} i M_{Trans} su konstante:

$$M_{base} = 0.05GB/hr, \tag{32}$$

$$M_{Trans} = 0.5 \tag{33}$$

Konačno, ukupni trošak budžeta se računa se izračunava sabiranje troškova procesora i memorije svake virtuelne mašine:

$$B_{(x)} = C_{(x)} + M_{(x)}, \tag{34}$$

gde $B_{(x)}$ označava ukupnu funkciju troškova budžeta krajnjeg korisnika, dok $C_{(x)}$ i $M_{(x)}$ označavaju funkcije troškova procesora i memorije, respektivno.

Uzimajući u obzir i vreme izvršavanja i troškove budžeta, podobnost se određuje na sledeći način:

$$H_{(x)} = MS_{(x)} + B_{(x)}, \tag{35}$$

gde $MS_{(x)}$ označava ukupno vreme izvršavanja svih zadataka (makespan), što u ovom modelu predstavlja funkciju performansi sistema, koje treba da bude manje ili jednako predefinisanom roku za izvršavanje zadatka. Vrednost MS se izračunava kao:

$$MS_{(x)} \leq \sum_{i=1}^{|T|} D_i, \tag{36}$$

gde je notacija D_i označava rok za izvršavanja zadatka T_i. U prethodnoj jednačini (35), $B_{(x)}$ predstavlja funkciju budžeta zadatka, koja obuhvata troškove procesora i memorije i koja treba da bude manja ili jednaka raspoloživom budžetu krajnjih korisnika.

Na osnovu toga, ograničenja funkcije troškova budžeta $B_{(x)}$ mogu da se modeluju kao:

$$B_{(x)} \leq \sum_{i=1}^{|T|} B_i, \tag{37}$$

gde B_i reprezentuje trošak budžeta krajnjeg korisnika zadatka T_i.

Kao što može da se zaključi iz jednačine (35), za rešavanje ovako formlisanog model višekriterijumske optimizacije korišćen je pristup težinske sume [72], gde oba formulisana cilja $MS(x)$ i $B(x)$ imaju jednake skalarne težinski koeficijent $\omega = 1$. Drugim rečima, oba cilja imaju isti prioritet.

Rezultati koji su ostvareni primenom metaheuristika rojeva za rešavanje ovog modela raspoređivanja poslova na klaudu publikovani su u [114].

5 Optimizacioni algoritmi

U opštem smislu, algoritam je procedura koja se izvršava korak po korak i koja se sastoji iz niza instrukcija i izračunavanja. Koraci izračunavanja zavise pre svega od određenog algoritma i od konteksta u kojem se primenjuje [157].

Optimizacioni algoritam je procedura (u pojedinim slučajevima iterativnog karaktera), gde se u svakoj iteraciji upoređuje kvalitet generisanih rešenja, i koja se izvršava sve dok se ne pronađe optimalno ili zadovoljavajuće rešenje, ili dok se ne zadovolji uslov prekida (eng. termination condition). Optimizacioni algoritmi su posebna grupa matematičkih algoritama, čiji je glavni zadataka da pronađu najbolju ili zadovoljavajuću alternativu rešenja problema u okviru određenih potencijalnih ograničenja.

Kada se rešava bilo koji optimizacioni problem, proces optimizacije uključuje optimizacioni algoritam, efikasni numerički simulator i realnu reprezentaciju fizičkih procesa koje je potrebno modelirati i optimizovati [156]. Proces optimizacije često zahteva puno vremena uz relativno visoke troškove izvršavanja. Nakon što se formuliše dobar model optimizacionog problema koji se rešava, troškovi izvršavanja zavise pre svega od primenjenog optimizacionog algoritma i efikasnosti korišćenog simulatora.

U literaturi se kao tri glavna izazova optimizacije koja se bazira na simulaciji najčešće navode [156]: efikasnost algorithma, preciznost i efikasnost numeričkog simulatora i izbor odgovarajućeg algoritma za određeni problem. Nažalost, ne postoji univerzalno pravilo, niti smernice koje bi obezbedile pravi odgovor na navedene izazove. Efikasnost optimizacionog algoritma zavisi od faktora poput principa na kojima algoritam funkcioniše, od ulaznih informacija koje su potrebne (na primer funkcija cilja i njeni izvodi) i od implementacionih detalja. S druge strane, efikasnost numeričkog simulatora zavisi od metoda (algoritama) koje se koriste za rešavanje problema i od složenosti samog problema. Konačno, ne postoji algoritam koji je efikasan za sve tipove problema, tj. važi teorema da besplatni ručak ne postoji (eng. No Free Lunch Theorem).

5.1 Kategorizacija optimizacionih algoritama

U savremenoj literaturi je dostupan veliki broj taksonomija optimizacionih algoritama, koje algoritme klasifikuju po osnovu raznih kriterijuma (karakteristika) koji se uzimaju o ubzir. Koja god klasifikacija da se koristi, najvažnije je da se izabere algoritam koji će da za zadati optimizacioni problem dati najbolje rezultate, uz napomenu da se ponekad najbolji rezultati postižu kombinacijom različitih algoritama.

Prema jednoj klasifikaciji [156], na osnovu kriterijuma da li se u procesu pretrage koriste, ili se ne koriste informacije o izvodu funkcije koja se optimizuje, optimizacioni algoritmi se dele na one koji koriste izvod (eng. derivate-based algorithms) i na one koji ne koriste izvod (eng. derivate-free methods). Neki od poznatijih predstavnika prve grupe su Gauss-Newton metod i metoda najbržeg spuštanja (eng. steepest descent method), dok se kao jedan od više korišćenih predstavnika algoritama koji ne koriste informacije o izvodu funkcije navodi Nelder-Mead simpleks metoda spuštanja niz brdo (eng. Nelder-Mead downhill simplex method).

Ako se u obzir uzme kriterijum mobilnosti, optimizacioni algoritmi mogu da se svrstaju u grupu algoritama lokalne ili globalne pretrage [156]. Lokalni algoritmi u većini slučajeva konvergiraju ka lokalnom optimumu (ponekada, ali retko i ka globalnom optimumu) i kao takvi nisu sposobni da „pobegnu" iz domena prostora pretrage gde se nalazi suboptimalno rešenje. Kao jedan od predstavnika optimizacionih algoritama lokalne pretrage navodi se jednostavni algoritam penjanja uz brdo (eng. simple hill-climbing algorithm). U slučaju problema globalne optimizacije, gde postoji više lokalnih i jedan ili više globalnih optimuma, potrebno je koristiti algoritme globalne pretrage. Jedna od važnijih karakteristika globalnih algoritama je slučajnost (eng. randomization). Tako na primer, ako se primeni jednostavna strategija, kojom se u određenim iteracijama izvršavanja algoritma penjanja uz brdo početno rešenje generiše slučajno, ovaj algoritam lokalne pretrage se transformiše u algoritam globalne pretrage.

Jedna od najšire korišćenih taksonomija optimizacionih algoritama deli algoritme na determinističke (eng. deterministic) i na stohastičke (eng. stochastic), tj. probabilističke (eng. probabilistic) [148]. Jedna druga podela koja je slična navedenoj, s tim što koristi malo drugačiju terminologiju, sve optimizacione algoritme razvrstava na egzaktne i na heurističke [61], gde se heuristike dalje razvrstavaju u kategorije

heuristika za specifične probleme i na metaheuristike. Egzaktni algoritmi odgovaraju determinističkim, dok heuristički u svakom koraku izvršavanja donose odluku o sledećem koraku u zavisnosti od kvaliteta trenutnih rešenja.

Osnovna karakteristika determinističkih algoritama je da pod bilo kakvim uslovima (na primer računarska platforma) i pri svakom izvršavanju na osnovu istih ulaza uvek generišu iste izlaze zato što uvek slede iste niz instrukcija (naredbi). Međutim, za probleme kod kojih relacije između kandidat rešenja i njihove podobnosti (eng. fitness) nisu očigledne, u slučaju kada je broj dimenzija prostora pretrage veliki i kada se optimizuju složenije funkcije sa više lokalnih optimuma, deterministički algoritmi nisu sposobni da generišu zadovoljavajuće suboptimalno rešenje zato što bi izvršavanje determinističkog algoritma u ovim slučajevima trajalo neprihvatljivo dugo. Drugim rečima, deterministički algoritmi ne mogu efikasno da se primene za rešavanje NP teških problema, jer bi bilo potrebno da se ispita svako potencijalno rešenje problema, kojih ima neograničeno mnogo.

Kao jedan od primera iterativnih determinističkih algoritama navodi se Newton-Raphson algoritam. Ovaj algoritam u iteraciji $t+1$ ima za cilj da generiše bolje rešenje x^{t+1} tako što unapređuje rešenje u x^t iz iteracije t. U nastavku je ovaj klasičan metod ukratko prikazan.

Newton-Raphson algoritam je često korišćena tehnika za optimizaciju funkcije tako što pronalazi nule (kritične tačke) funkcije koja se optimizuje. Neka je kritična tačka (nula) funkcije $H : \mathbb{R}^n$ u tački x^*, tako da važi $H(x^*) = 0$. Primenom ovog algoritma x^* se izračunava u iteracijama u odnosu na početnu slučajnu vrednost x^0 pomoću sledećeg izraza [89]:

$$x^{t+1} = x^t - DH(x^t)^{-1}H(x^t) \tag{38}$$

Jednačina (38) dobija se linearnom aproksimacijom jednačine $y = H(x)$:

$$\Delta y = DH(x)\Delta x \tag{39}$$

Jednačinom (39) se predviđa promena Δy vrednosti funkcije $H(x)$ u odnosu na promenu Δx vrednosti x. Obzirom da se traže nule funkcije H, postavlja se da je $\Delta y = -H(x)$ i jednačina (39) se rešava po Δx:

$$\Delta x = -DH(x)^{-1}H(x) \tag{40}$$

Zamenom vrednosti $x = x^t$ i $\Delta x = x^{t+1} - x^t$, izraz (38) dobija se pomoću jednačine (40). Obzirom da je izraz (39) samo aproksimacija, ne očekuje se da će odmah nakon prvog izvršavanja algoritma uslov $H(x^{t+1}) = 0$ biti zadovoljen, ali postoje razumna očekivanja da će primenom jednačine (38), u više iteracija, generisati sve bolja i bolja aproksimacija tačke x^*.

Analizom Newton-Raphson algoritam vidi se da jednačina (39) postiže lošu aproksimaciju kada su vrednosti Δx i Δy velike. Dakle, zaključuje se da je će algoritam garantovano da konvergira samo kada je početna vrednost x^0 blizu optimalne vrednosti x^*. U slučaju da je x^0 daleko od x^*, algoritam možda nikada neće konvergirati ka optimumu.

Brzina konvergencije rešenja x^t može da se testira primenom sledećeg izraza [89]:

$$|x^{t+1} - x^t| < \varepsilon, \tag{41}$$

za neku ε vrednost koja je veća od 0. Međutim, ovakav test je osetljiv na promene skaliranja vrednosti x^t i zbog toga se najčešće koristi test gde se uzimaju u obzir apsolutne i relativne vrednosti tolearncije, koje se označavaju sa E_a i E_r, respektivno:

$$|x^{t+1} - x^t| < E_r|x^t| + E_a \tag{42}$$

Kasnih 50-tih godina 20. veka prvi put su počele da se koriste deterministčke metode za rešavanje problema globalne optimziacije koje se oslanjaju na princip "podeli, pa vladaj"(eng. divide-and-conquer). Primenom ovih metoda, glavni problem se deli na više manjih problema (podproblema), čime se značajno smanjuje njihova složenost. Na taj način, smanjivanjem složenih teorijskih struktura, ove metode se u procesu istraživanja domena pretrage oslanjaju na intenzivna računarska izračunavanja.

Kao jedna od najšire primenjivanih determinističkih metoda iz ove grupe navodi se algoritam grananja i ograničavanja (eng. branch and bound - BB), koji glavni problem deli na grane (podprobleme). Pretragom literature može da se ustanovi da se BB algoritmi i danas široko primenjuju u raznim domenima optimizacije, poput bežičnih senzorskih mreža [77], energetskih sistema [162], više-kriterijumske

optimizacije [93], kao i u domenu ostalih praktičnih problema [86], [3].

Nasuprot determinističkim, izvršavanje algoritama koji pripadaju grupi stohastičkih optimizacionih metoda, bazira se na slučajnosti. Ovi algoritmi u svakom koraku izračunavanja biraju sledeći korak pod uticajem slučajnog (pseudo-slučajnog) elementa. Drugim rečima, stohastički algoritam u svakom puštanju (izvršavanju) prati različite putanje. Slučajan izbor se simulira primenom generatora pseudo-slučajnih sekvenci. Pomoću teorije verovatnoće izvode se dokazi za konvergenciju stohastičkih algoritama.

U kategorizju stohastičkih optimizacionih algoritama pripada veliki broj pristupa, kao što metod Monte Karlo, metaheuristike inteligencije rojeva,

Veliki broj optimizacionih algoritama i metoda pripada grupi stohastičkih optimizatora, kao što su Monte Karlo metod i algoritmi koji su nastali iz ovog metoda, evolutivni algoritmi, algoritmi inteligencije rojeva, itd. Sledeće poglavlje posvećeno je ovoj temi.

Zbog značaja za ovu doktorsku disertaciju, u nastavku su prikazane heurističke i metaheurističke optimizacione metode i algoritmi.

5.2 Heuristike

Heuristike (od starogrčke reči *heuriskein*), označavaju grupu empirijskih, iskustvenih metoda za rešavanje problema. Tokom procesa pretrage, heurističke metode pokušavaju da pronađu što bolje rešenje, ali ne mogu da garantuju da će pronađeno rešenje biti najbolje (optimalno) [73].

Za neke probleme, posebno one koje pripadaju grupi NP teških problema, moguće je primeniti egzaktne (determinističke) optimizacione metode i algoritme, koje garantuju pronalazak optimalnog rešenja, ali bi u ovom slučaju njihova primena bila nepraktična, jer bi vreme izvršavanja trajalo neprihvatljivo dugo. Zbog toga se u ovakvih slučajevima primenjuju heuristike, koje na garantuju pronalazak optimalnog rešenja, ali zato u većini slučajeva heuristike mogu da generišu zadovoljavajuća (suboptimalna) rešenja u prihvatljivom vremenskom periodu. Heuristike se primenjuju za rešavanje konkretnih i specifičnih problema, koristeći individualna svojstva samih problema.

O značaju koje heuristike imaju u savremenom računarstvu, operacionim istra-

živanjima i drugim naučnim granama svedoči veliki broj naučnih publikacija koje se njima bave, kao i veliki broj vrhunskih naučnih časopisa čija su glavna tema heuristike. Primer je poznati časopis *Journal of Heuristics* izdavača *Springer*. Osim toga, svake godine se organizuje i veliki broj međunarodnih konferencija na kojima se prikazuju teme koje su vezane za heurističke i metaheurističke algoritme, kao što je na primer *Metaheuristic International Conference (MIC)*.

Osim glavnog razloga koji je doveo da nastanka i razvoja heurističkih algoritama - nesposobnost determinističkog algoritma da u prihvatljivom vremenskom periodu generiše optimalno ili zadovoljavajuće (suboptimalno) rešenje, kao neki od dodatnih argumenata za razvoj heurističkih metoda navode se sledeći [73]:

- za određeni problem nije dostupna tehnika koja može da ga optimizuje;
- nemogućnost da se raspoloživi egaktni algoritam primeni na dostupnim hardverskim resursima;
- veća fleksibilnost heuristika u odnosu na determinističke metode u kontekstu mogućnosti modeliranja dodatnih svojstava praktičnog problema koji se optimizuje i
- prostor za inkorporaciju heurističkog algoritma u metodu pretrage globalnog algoritma, koji garantuje pronalaženje optimalnog rešenja određenog problema.

Prema [73], dobar heuristički algoritam treba da sadrži sledeće karakteristike:

- generisanje rešenja treba da se izvrši u prihvatljivom vremenskom periodu;
- maksimizovanja verovatnoće dobijanja rešanja koje je blisko optimalnom (zadovoljavajuće rešenje) i
- minimizacija verovatnoće generisanja lošeg rešenja.

Dostupne heurističke metode su heterogene i zbog toga je teško definisati sveobuhvatnu taksonomiju ovih metoda. Zbog ove činjenice, u savremenoj literaturi je dostupan veliki broj klasifikacija heurističkih metoda i algoritama [124]. Prema jednoj široko korišćenoj kategorizaciji, mogu da se izdvoje sledeće grupe heurističkih

metoda [73]: induktivne, redukcione, dekompozicione, konstruktivne i metode lokalne pretrage.

Induktivne heuristike (eng. inductive heuristics) vrše generalizaciju jednostavnih i manjih verzija problema polazeći od toga da tehnike koje mogu efikasno da se primene na manje verzije problema, mogu isto tako da se primene i na ceo problem.

Identifikaciju svojstava dobrih rešenja problema i njihovo uvođenje u proces optimizacije u formi granica problema predstavljaju osnovne funkcionalne principe redukcionih heuristika (eng. reducation heuristics). Na ovaj način se pojednostavljivanjem polaznog problema prostor pretrage ograničava uz rizik da će optimalna rešenja biti izostavljena.

Dekompozicione heuristike (eng. decomposition heuristics) funkcionišu po osnovu principa "podeli, pa vladaj", tako što dele glavni problem na manje podprobleme, koji pripadaju klasi glavnog problema, i koji se lakše mogu optimizovati.

Na osnovu filozofije izgradnje rešenja od temelja u fazama, po principu korak po korak, funkcionišu konstruktivne heuristike (eng. constructive heuristics). Ovakvi pristupi se uglavnom koriste za optimizaciju standardnih kombinatornih problema.

Heuristike lokalne pretrage (eng. local search heuristics) na početku izvršavanja generišu dopustivo rešenje problema, koje zatim u iterativnom procesu pokušavaju da poprave. U svakoj iteraciji procesa pretrage, rešenja se pomeraju ka drugim rešenjima koja su većeg kvaliteta (imaju bolju vrednost funkcije podobnosti).

Sa ciljem poboljšanja brzine konvergencije procesa pretrage, heurističke metode se najčešće ugrađuju u metaheurističke algoritme. Na osnovu pretrage raspoložive literature, može da se vidi da se kao komponente metaheurističkih pristupa najčešće koriste konstuktivne i heuristike lokalne pretrage.

5.3 Metaheurisitke

Zbog efikasne primene i rezultata koje generišu heurističke metode za rešavanje raznovrsnih praktičnih problema, posebno u poslednje dve decenije poraslo je interesovanje za ubrzani razvoj i unapređenje heuristika, gde su kao rezultat nastale metaheuristike, tj. metaheuristički algoritmi (eng. metaheuristic algorithms). U nazivu metaheuristika, slično kao i u slučaju termina heuristika, prefiks *meta* takođe predstavlja grčku reč, koja označava metodologiju višeg nivoa.

Prema jednom stanovištu, metaheuristike obuhvataju generičke skupove pravila koja mogu da se primene na rešavanje velikog broja optimizacionih problema. Suštinski principi metaheuristika oslanjaju se na opšte optimizacione algoritme koji u iteracijama pokušavaju da poprave trenutno rešenje.

Prema [124] metaheuristika se definiše kao skup algoritamskih koncepata koji se primenjuju za definisanje heurističkih metoda koje se koriste za širok dijapazon problema. Jednostavnije, metod za pronalaženje odgovarajuće (dobre) heuristike za specifičan problem predstavlja metaheuristiku. Metaheuristički algoritmi i metode našli su primenu u raznim oblastima, kao što su problemi inženjerskog dizajna u elektronici, finansijama, fizici, hemiji, biologiji, medicini, itd.

Prema pojedinim izvorima, termin metaheuristika je prvi upotrebio Fred Glover 1986. godine [39]. Obično se za heurističke metode kaže da izvršavaju proces pretrage primenom principa "pokušaja i greške" (eng. trial and error), dok s druge strane metaheuristike kao "strategije višeg nivoa" usmeravaju i modifikuju heurističke procedure kako bi se generisala inovativna rešenja, čiji kvalitet prevazilazi lokalni optimum [160].

Prilikom izbora najbolje (zadovoljavajuće) heuristike za specifičan problem, metaheuristički pristupi pokušavaju da odgovore na sledeće zahteve [124]:

- koje vrednosti parametara daju dobre rezultate kada se primeni heuristika x na problem p;
- na koji način je moguće prilagoditi parametre heuristike x da bi se dobila bolja rešenja problema p;
- da li je za rešavanje problema p bolja primeniti heuristiku x ili heuristiku y.

Pregledom literature vidi se da postoji veliki broj metaheuristika koje ne postižu zadovoljavajuća rešenja. Da bi metaheuristička metoda bila što efikasnija i da bi generisala što bolje rezultate optimizacionih problema, potrebno je da poseduje sledeća svojstva [124]:

- jednostavnost. Metaheuristike treba da se baziraju na intuitivnim i lako razumljivim principima;

- preciznost. Svi koraci koje izvršava metaheuristički algoritam treba da budu precizno formulisani matematičkim pojmovima i terminima;
- doslednost. Sve faze u procesu primene algoritma treba da budu konzistentne sa opštim pravilima u skladu sa kojima je metaheuristika formulisana;
- efikasnost. Kada se metaheuristika primeni na konkretan problem iz realnog okruženja treba da generiše rešenja koja su bliska optimalnom. Ovo se posebno odnosi na slučaj kada se metaheuristika primeni na zvanične test primere (eng. benchmark problems);
- efektivnost. Za svaki konkretan problem algoritam treba da ostvari optimalno, ili makar zadovoljavajuće rešenje u razumnom vremenskom periodu i
- univerzalnost. Metaheuristička metoda treba da se bazira na što opštijim osnovnim principima i pravilima, kako bi mogla da se sa lakoćom, i uz minimalne modifikacije, primeni i na druge problema.

Slično kao i u slučaju heurističkih metoda, s obzirom na heterogenost metaheurističkih algoritama, teško je dati kategorizaciju koja bi obuhvatala sve dostupne metaheuristike. Prema jednoj klasifikaciji, metaheuristike se dele na one koje koriste skup struktura okoline i na one koje koriste jednu strukturu okoline. U svetskoj literaturi uglavnom se koriste optimizacioni algoritmi koji koriste jednu strukturu okoline, ali postoje i metode koje koriste skup struktura okoline, kao što je tehnika promenljivih okolina.

Jedna od "najgrubljih" podela metaheuristika može biti na determinističke i stohastičke. Slično kao i u slučaju kategorizacije optimizacionih algoritama, determinističke metaheuristike za jednake početne uslove uvek daju jednake rezultate, dok stohastičke metaheuristike u procesu pretrage koriste slučajna pravila, tako da za iste početne uslove mogu da generišu različita krajnja rešenja.

Prema još jednoj podeli, metaheuristike se dele na poboljšavajuće, konstruktivne i hibridne. Kao jedan od predstavnika poboljšavajućih metaheuristika navodi se metoda simuliranog kaljenja (eng. simulated annealing), koja na početku izvršavanja bira slučajno početno rešenje, koje se zatim popravlja u iterativnom procesu. Konstuktivne metaheuristike, poput konstuktivnih heuristika grade rešenje problema sledeći

princip „korak po korak". Poznatiji predstavnik ovakvih metaheuristika je pohlepni algoritam (eng. greedy algorithm). Konačno, u poslednje vreme su sve popularnije hibridne metaheuristike, koje nastaju kombinovanjem dva ili više algoritama, tako što u hibridno rešenje inkorporiraju prednosti jednih uz istovremeno eliminisanje nedostataka drugih algoritama. S obzirom da je za rešavanje praktičnog problema u ovoj tezi korišćen jedna hibridna metaheuristika, kategorija hibridnih rešenje je u ovom slučaju od posebnog značaja.

Prema jednoj od najsveobuhvatnijih podela metaheurističkih algoritama, koja je data u [124], metaheuristike se dele na:

- determinističke i stohastičke;
- one koje se baziraju na populaciji rešenja (eng. population-based) i koje se baziraju na jednom rešenju (eng. single solution-based);
- one koje koriste memoriju (eng. memory-based) i na one koje ne koriste memoriju (eng. memory-less) i
- metaheuristike koje su inspirisane prirodom (eng. nature-inspired) i metaheuristike koje nisu inspirisane prirorodom (eng. nonnature-inspired).

Metaheuristike koje se baziraju na populaciji rešenja u procesu pretrage simultano (pseudo-paralelno ili paraleleno, ako se izvršavaju na sistemi za vrše procesora i ako su implementirani primenom paradigme konkurentnog programiranja) tansformišu i manipulišu sa više potencijalnih rešenja. Populacione metaheuristike su uglavnom orijentisane na diversifikaciji, tj. na ispitivanju neistraženih delova prostora pretrage. S druge strane, metaheuristike koje koriste samo jedno proces pretrage izvode manipulisanjem i transformisanjem tog jednog rešenja. Ove metode uglavnom koriste mehanizam eksploatacije u izvršavanju procesa pretrage. U praksi se ove dve vrste metaheursitika uglavnom koriste zajedno, zbog komplementarnih i kompatibilnih svojstava. Metaheuristike inteligencije rojeva, koje su od posebnog značaja za ovu tezu spadaju u grupu populacionih metaheuristika.

Osnovna razlika između metaheuristika koje koriste i koje ne koriste memoriju je u sposobnosti pamćenja (skladištenja) prethodno generisanih rešenja (rešenja koja su generisana u prethodnim iteracijama izvršavanja algoritma). Algoritmi koji ne koriste

memoriju, za razliku od algoritama koji se baziraju na memoriji, nisu u stanju da dinamički u toku izvršavanja ekstrahuju informacije koje su relevantne za izvršavanje procesa pretrage. Jedan od predstavnika metaheuristika koje ne koriste memoriju je simulirano kaljenje, dok metoda tabu pretrage pomoću mehanizama dugoročne i kratkoročne memorije ima mogućnost da pampti bitna svojstva rešenja koja su generisana u prethodnim iteracijama.

U slučaju metaheuristika koje su inspirisane prirodom, mehanizam kojim se usmerava i vodi proces pretrage (jednačina pretrage) inspirisan je prirodnim fenomenom. Kao dva najčešće korišćena prirodna fenomena navode se proces evolucije živih organizama i koolektivna inteligencija grupa (populacija) individua u prirodi, poput jata riba i ptica, kolonije mrava i pčela, grupa slepih miševa, itd. S druge strane, u slučaju metaheuristika koje nisu inspirisane prirodom, procedura koja vodi proces pretrage simulira proces čije poreklo ne datira iz prirode. Takođe je potrebno naglasiti da fenomen koji se simulira u slučaju prirodom inspirisanih metaheuristika može da bude i fizički proces, kao što je slučaj sa metaheuristikom simuliranog kaljenja.

Korisno je da se napomene da većina dostupnih metaheuristika ne pripada samo jednoj, već spada u više navedenih kategorija. Za istraživanje koje je sprovedeno za potrebe ove doktorske disertacije, posebno su značajne metaheuristike koje su inspirisane prirodom, u zbog toga je u nastavku detaljnije prikazana kategorizacija metaheuristika na osnovu kriterijuma da li je fenomen koji simulira proces pretrage inspirisan, ili nije inspirisan prirodnim mehanizmima.

5.3.1 Metaheuristike koje nisu inspirisane prirodom

Zajednička karakteristika svih metaheurističkih metoda jeste da su metaheuristike kompleskni sistemi koji se uglavnom sastoje iz heuristika, koje se baziraju na populaciji rešenja, u kombinaciji sa drugim metodama. Takođe, svi metaheuristički algoritmi tokom izvršavanja procesa pretrage uspostavljaju kompromis između globane, koja je načešće nasumična i lokalne pretrage [160].

Kao što je već i naglašeno, kao jedan od kriterijuma za podelu metaheurističkih algoritama navodi se priroda mehanizma koji usmerava i kontroliše proces pretrage. Jedna od glavnih karakteristika metaheuristika koje nisu inspirisane prirodom jeste da mehanziam koji izvodi pretragu ne modelira prirodne procese, niti grupu organizama

koji se mogu naći u prirodi.

U savremenoj literaturi je poznat širok dijapazon metaheuristika koje nisu inspirisane prirodnim fenomenon. Neki od ovih algoritama koriste Monte Karlo metode, dok pojedini koriste gramzive algoritme i lokalnu pretragu. Kao poznatiji predstavnici ove grupe algoritama mogu da se navedu diferencijalna evolucija i tabu pretraga.

Metod Monte Karlo (eng. Monte Carlo) pripada kategoriji stohastičkih optimizacionih metoda i kao takav tokom izvršavanja procesa pretrage koristi određeni nivo slučajnosti. Takođe, ovaj metod je poznat i kao jedan od najjednostavnijih dostupnih optimizatora. Osnovni principi funkcionisanja Monte Karlo metode su jednostavni. Iz skupa potencijalnih rešenja problema bira se "slučajno" rešenje i računa se njegov kvalitet, koji se u najvećem broju slučajeva izražava merom koja se naziva funkcija podobnosti (eng. fitness function). Ako je kvalitet slučajno izabranog rešenja veći od kvaliteta prethodno sačuvanog najboljeg rešenja, novoizabrano rešenje se čuva i proglašava se za novo trenutno najbolje rešenje. Nakon toga se bira novo test rešenje i čitav proces se ponavlja sve dok predefinisani kriterijum prekida izvršavanja algoritma nije zadovoljen [127]. Kriterijumi za prekid izvršavanja algoritma mogu biti različiti, od broja iteracija, do broja pokušaja da se unapredi trenutno najbolje rešenje.

Monte Karlo je relativno spor metod, zato što je potrebno da se testira veliki broj potencijalnih rešenja pre nego što se isgeneriše kvalitetno rešenje. Zbog toga se ovaj metod u velikom broju praktičnih implementacija koristi isključivo u kombinaciji sa nekom drugom procedurom koja umanjuje uticaj faktora slučajnosti kada se bira naredno rešenje koje će biti evaluirano.

Zbog činjenice da je Monte Karlo jedna od najpoznatijih stohastičkih optimizacionih metoda i zbog toga što je u literaturi poznat veliki broj njegovih aplikacija, u nastavku je prikazan pseudo-kod ovog algoritam.

Algoritam 1 Metod Monte Karlo

Faza inicijalizacije. Izbor početnog rešenja x^0 i proglašavanje da je $x^* = x^0$ i da je $f(x^*) = f(x^0)$
while nije postignut uslov završetka algoritma **do**
  Pokušaj. Selekcija slučajnog rešenja x' iz prostora rešenja X
  Evaluacija rešenja.
  if $f(x') < f(x^*)$ **then**
    $x^* = x', f(x^*) = f(x')$
  end if
end while

U navedenom pseudo-kodu, x^*, x^0 i $x^{'}$ označavaju redom najbolje, početno i novo rešenje, dok se notacije $f(x^*)$, $f(x^0)$ i $f(x^{'})$ odnose na funkciju podobnosti (kvalitet) najboljeg, početnog i novog rešenja, respektivno.

Metoda lokalne pretrage spada u grupu iterativnih optimizacionih tehnika, gde se u svakoj iteraciji proces pretrage usmerava od jednog do drugog rešenja, dokle god uslov završetka algoritam nije ispunjen, ili dokle god optimalno rešenje nije pronađeno. Na početku se bira slučajno početno rešenja, da bi se zatim ispitivala rešenja u njegovom susedstvu (okolini). S obzirom da se u okolini svakog rešenja nalazi više potencijalnih rešenja, potrebno je da se odredi i način, tj. mehanizam na koji se bira rešenje iz susedstva trenutnog rešenja. Neki od mogućih mehanizama izbora rešenja iz okoline trenutnog rešenja uključuju varijacije metode penjanje uz brdo, kao što su jednostavno penjanje uz brdo (eng. simple hill climbing), stohastičko penjanje uz brdo (eng. stohastics hill climbing) i penjanje uz brdo najstrimijom putanjom (eng. steepest accent hill climbing).

Metode lokalne pretrage se retko koriste izolovano, već se uglavnom koriste u kombinaciji sa drugim metodama zbog toga što se njihov proces pretrage zaustavlja čim naiđu na prvi lokalni minimum, koji nije a priori i globalni optimum. Međutim, u literaturi su poznate metode lokalne pretrage koje u svakoj iteraciji pretragu počinju od novog, slučajno izabranog početnog rešenja. Kao jedan od predstavnika ovakvih metoda navodi se više-startno lokalno pretraživanje (eng. multi-start local search - MLS).

Algoritam koji smanjuje prostor pretrage tako što pretragu usmerava ka trenutno najboljem rešenju naziva se gramzivi algoritam (eng. greedy algorithm). Kao glavna prednost ovog algoritma navodi se što se na ovaj način kompleksnost samog optimizacionog procesa smanjuje. Međutim na ovaj način gramzivi algoritam nije uvek u stanju da pronađe globano optimalno rešenje. Zbog ovoga, gramzivi algoritmi, koji se odnose na širi skup algoritamskih koncepata koji se baziraju na "gramzivosti" se uglavnom koriste upareni sa drugom heuristikom ili metaheuristikom, gde navode proces pretrage ka trenutno najboljem rešenju u populaciji [46].

S obzirom na široku primenu i rasprostranjenost u svetskoj literaturi, kao poznatiji predstavnici metaheuristika koje nisu inspirisane prirodom navode se tabu pretraga (eng. taboo search - TS) i diferencijalna evolucija (eng. differential evolution - DE).

Metaheuristika TS, koju je kreirao F. Glover 1986. godine [40], spada u grupu metoda lokalne pretrage koje koriste memoriju za čuvanje informacija o generisanim rešenjima. Primenom TS algoritma, nedostaci metoda lokalne pretrage, koji se odnose na mogućnost da se algoritam zaglavi u nekom od suboptimalnih domena prostora pretrage, se eliminišu uvođenjem kategorije tabu rešenja. Atributom "tabu" označavaju se rešenja koja su evaluirana u nekoj od prethodnih iteracija izvršavanja algoritma, ili ako ne zadovoljavaju neko od ograničenja ili pravila definisanih u matematičkoj formulaciji problema koji se optimizuje. Ova metoda generiše listu od poslednjih m tabu rešenja. Sam proces pretrage se izvršava tako što se bira rešenje koje se nalazi u okolini trenutnog rešenja i koje se ne nalazi u listi tabu rešenja, čak i ako je podobnost izabranog rešenja manja od podobnosti trenutnog rešenja, čija se okolina ispituje.

Metaheuristika DE, čiji su tvorci Storm i Price 1997. godine [104], pokazala se kao efikasna metoda u rešavanju problema globalne optimizacije. Ovaj algoritam koristi operatore ukrštanja i mutacije koje generišu nove jedinke na osnovu postojećih jedinki u populaciji. Takođe, DE primenjuje elitističku selekciju, gde se u svakoj iteraciji najbolje jedinke iz populacije zadržavaju i prenose se u sledeću generaciju. Kao jedna od glavnih prednosti metaheuristike DE navodi se korišćenje relativnog malog broja kontrolnih parametara. Glavne mane ovog algoritma su stagnacija procesa pretrage i spora konvergencija [160]. U literaturi su poznate brojne aplikacije metaheuristike DE [32].

Od ostalih pripadnika grupe metaheuristika koje nisu inspirisane prirodom korisno je spomenuti metodu promenljivih okolina eng. variable neighborhood search – VNS) [83] i metoda raspršene pretrage eng. scatter search) [38].

5.3.2 Metaheuristike koje su inspirisane prirodom

Glavna karakteristika metaheuristika inspirisanih prirodom ogleda se u tome što mehanizam koji izvršava proces pretrage simulira neki prirodni fenomen. Taj prirodni fenomen je naječešće proces biološke evolucije, ili grupa organizama iz prirode, gde članovi grupe (individue) komuniciraju na indirektan način i time manifestuju inteligentno ponašanje. Kao jedan od primera navodi se kolonija mrava, gde mravi pronalaze najkraću putanju između svoje kolonije i izvora hrane ostavljanjem

supstance koja se naziva feromon.

Na osnovu kriterijuma tipa prirodnog fenomena kojeg simuliraju, metaheuristike inspirisane prirodom dalje mogu da se kategorizuju u dve grupe. U prvu grupu spadaju pristupi evolutivnih algoritama (eng. evolutionary algoritms - EA), koji simuliraju proces biološke evolucije, dok algoritmi koji pripadaju drugoj kategoriji, metaheuristike inteligencije rojeva (eng. swarm intelligence), modeliraju kooperativno inteligentno ponašanje grupe individua koje se mogu naći u prirodi. Evolutivni algoritmi u literaturi su još poznati i pod nazivom algoritmi evolutivnog izračunavanja (eng. evolutinary computation - EC).

Teorijski principi na kojima se funkcionisanje navedenih kategorija metaheuristika inspirisanih prirodom bazira su slični, ali ipak između ove dve grupe postoje suštinske razlike. U literaturi je poznat veliki broj hibridnih metaheuristika, koje eliminišu nedostatke jednih inkorporiranjem prednosti drugih pristupa i takvi hibridni pristupi često kombinuju metaheuristike inteligencije rojeva sa genetskim operatorima evolutivnih algoritama. Primeri takvih hibrida mogu se naći u [119], [116], [136], [12].

Teorijska pozadina EA data je u zakonima nasleđivanja koje je definisao Gregor Mandel i u teoriji evolucije, čiji je tvorac Čarls Darvin [30]. Suština je relativno jednostavana - prirodne vrste tokom vremena evoluiraju, gde se ukrštanjem roditelja dobijaju potomci, koji su osim što nasleđuju karakteristike (eng. traits) roditelja, podvrgnuti i slučajnoj mutaciji. Vremenom se individue adaptiraju (prilagođavaju) okolini u kojoj žive, čime dolazi do promene njihovog genetskog materijala. Primenom mehanizma prirodne selekcije, samo najjače individue opstaju i generišu potomke. Na taj način, iz generacije u generaciju, jedinke postaju sve robusnije i kvalitetnije. Slično može i da se primeni u procesu pretrage EA.

Algoritmi iz familije EA su populacioni, što znači da se proces pretrage izvršava modeliranjem grupe individua. Ovi algoritmi su takođe i iterativni, gde se u uzastopnim iteracijama generišu potencijalno bolja rešenja primenom osnovnih genetskih operatora ukrštanja (eng. crossover) i mutacije (eng. mutation). Individue koje se ukrštaju (roditelji) će imati jednog ili više potomaka, što zavisi od samog kvaliteta individua i od samog algoritma. Veliki broj ovih metaheuristika koristi elitističku selekciju (eng. elitist selection), što znači da će samo one jedinke koje su se najbolje adaptirale promenama u okruženju (u ovom slučaju jedinke koje imaju

najveću vrednost funkcije podobnosti), da "prežive" i da pređu u sledeću iteraciju (generaciju) izvršavanja algoritma. Na ovaj način u skladu sa teorijom genetskog nasleđivanja Georga Mandela, vrši se propagacija najboljeg genetskog materijala u sledeću generaciju.

Pseudo-kod jednostavnog EA dat je u Algoritmu 2.

Algoritam 2 Pseudo-kod jednostavnog EA

```
generiši slučajnu populaciju
repeat
    selekcija roditelja
    primena operatora ukrštanja nad roditeljima i kreiranje potomaka
    primena operatora slučajne mutacije nad potomcima
    izračunavanje kvaliteta (podobnosti) potomaka
    izbor jedinki za sledeću generaciju primenom operatora selekcije
until nije zadovoljen uslov završetka izvršavanja algoritma
kraj
```

Najstariji i najpoznatiji predstavnici familije EA su genetski algoritmi (eng. genetic algorithms - GA).

Otac GA je Džon Holand, koji je prvi put predstavio ovu grupu algoritama u svojoj knjizi "Adaptacija u prirodnim i veštačkim sistemima: uvodna analiza i primene u biologiji, kontroli sistema i veštačkoj inteligenciji " [43]. Algoritmi koji spadaju u GA pristupe koriste populaciju jedinki, koje predstavljaju tačke u prostoru istraživanja, tj. potencijalna rešenja. Populacija jedinki se u većini praktičnih implementacija najčešće sastoji od 10 do 200 veštačkih agenata. Pre samog procesa optimizacije, potrebno je sva potencijalna rešenja (individue) na određeni način kodirati, a način kodiranja zavisi od prirode problema koji se rešava. Jedinke se najčešće kodiraju kao binarni nizovi fiksne dužine, ali postoje i slučajevi kada se koristi kodiranje slovima engleskog alfabeta. Zbog činjenice da ne postoji univerzalni "recept", kodiranje je jedan od najvećih izazova u primeni GA pristupa.

Kao i ostali EA, GA pretragu sprovode tako što koriste operatore ukrštanja i mutacije. Ukrštanjem se kombinuju bolja potencijalna rešenja iz populacije i na taj način pretraga konvergira ka optimalnom domenu. S druge strane, operatorom mutacije se slučajno modifikuje jedna ili više komponenti rešenja. Ovaj operator se koristi kako bi se predupredio scenario da se proces pretrage zaglavi u nekom od suboptimalnih domena prostora koji se istražuje.

Individue (jedinke) se u kontekstu GA prikazuju kao hromozomi, a njihovi delovi

se nazivaju geni. Tako na primer, ako se optimizuje neka funkcija, funkcija sa svim parametrima predstavlja hromozom, dok pojedinačni parametri modelriaju gene. Za svaku individu u populaciji se računa kvalitet, koji je predstavljen funkcijom podobnosti. Operatorom selekcije se biraju najbolje jedinke u populaciji, koje učestvuju kao roditelji u sledećoj generaciji izvršavanja algoritma. Individue koje imaju lošije vrednost funkcije podobnosti imaju manju šansu za reprodukciju u narednoj generaciji i postepeno izumiru.

Prikaz jednostavnog (osnovnog) GA dat je u Algoritmu 3.

Algoritam 3 Pseudo-kod jednostavnog GA

```
kreiranje pseudo-slučajne početne populacije od n hromozoma (x) dužine m
repeat
   evaluacija podobnosti fit(x) svih hromozoma iz populacije
   while n potomaka se ne kreira do
      selekcija dva hromozoma roditelja iz trenutne populacije
      ukrštanje izabranih roditelja sa verovatnoćom p_c i kreiranje dva potomka
      sa verovatnoćom p_m primena operatora mutacije nad potomcima
   end while
   zamena trenutne populacije individuama iz nove populacije
until nije dostignut maksimalni broj generacija izvršavanja algoritma
```

U literaturi može da se nađe veliki broj primena GA za razne optimizacione probleme [74], [42], [99].

Od ostalih predstavnika familije EA korisno je spomenuti genetsko programiranje (eng. genetic programming - GP), čiji je tvorac John Koza [60]. Algoritam GP je zapravo tehnika za evoluiranje računarskih programa, gde je cilj da se pronađe optimalan ili zadovoljavajući program na osnovu definisanih kriterijuma pretrage. Sam proces pretrage ove metode podseća na tehniku penjanja uz brdo.

Osim algoritama iz navedene dve grupe EA, postoje i drugi pristupi koji ne spadaju ni u jednu kategoriju. Tako na primer, metaheuristika simulirano kaljenje (eng. simulated annealing - SA) spada u grupu metaheuristika inspirisanih prirodom, ali ne i u dve navedene kategorije. Pristup SA emulira fizički proces zagrevanja i kontrolisanog hlađenja metala do stanja kristalne rešetke sa većom veličinom kristala i najmanjom energijom [47].

S obzirom da su metaheuristike inteligencije rojeva jedna od glavnih oblasti ove doktorske disertacije, njima je posvećeno celo naredno poglavlje.

6 Metaheuristike inteligencije rojeva

U ovoj glavi prvo je definisan pojam i prikazani su osnovni principi metaheuristisika inteligencije rojeva. Zatim su detaljno opisane osnovne implementacije algoritma koji su unapređeni u prikazanom istraživanju. Na kraju glave dat je pregled literature, gde su prikazane primene algoritama rojeva na probleme raspoređivanja poslova i balansiranja opterećenja na klaudu, ali i na druge praktične NP teške probleme, čiji su rezultati objavljeni u literaturi.

6.1 Pojam i osnovni principi metaheuristika rojeva

Metaheuristike inteligencije rojeva (eng. swarm intelligence metaheuristics) spadaju u grupu metaheuristika inspirisanih prirodom, što znači mehanizam (procedura) koja izvršava pretragu simulira prirodni fenomen. U slučaju algoritama rojeva, procedura pretrage simulira grupu organizama, poput jata ptica i riba, kolonije mrava, rojeve pčela, krda slonova, grupe slepih miševa, itd.

Istraživanjem grupe organizama iz prirode, uočeno je da grupa (roj) individua ispoljava neki vid inteigentnog ponašanja, bez obzira što se sama grupa sastoji iz relativno nesofistiricanih i jednostavnih jedinki. Takođe, uočeno je da jedinke u grupi razmenjuju informacije uspostavljanjem nekog vida indirektne komunikacije, što omogućava da grupa funkcioniše kao celina i da teži ka jednom cilju. Zajednički cilj može da bude pronalaženje hrane, izbegavanje opasnosti, itd. Navedene činjenice su bili osnovni motivacioni faktori za nastanak i razvoj algoritama inteligencije rojeva.

Roj ili jato čini veliki broj homogenih i prostih jedinki, koje međusobno komuniciraju i uspostavljaju interakcije sa svojim okruženjem, a pri tom ne postoji centralna komponenta koja sinhronizuje njihovo ponašanje. Zbog činjenice da simuliraju jato ili roj, metaheuristike inteligencije rojeva spadaju u kategoriju populacionih algoritama, koji inkrementalno u iteracijama izvršavaju proces pretrage, sa ciljem pronalaska optimalmnog, ili približno optimalnog rešenja. Zbog svojih karakteristika, algoritmi rojeva su dokazali da mogu brzo da generišu kvalitetna rešenja heterogenih praktičnih problema [59].

Algoritmi rojeva pripadaju domenu veštačke inteligencije (eng. artificial intelligence - AI), i kao takvi predstavljaju relativno mladu granu ove perspektivne

naučne oblasti. Jedna od važnijih karakteristika metaheuristika rojeva je mogućnost komunikacije i razmene informacija između individua. U opštem smislu, komunikacija i interakcije između članova roja mogu da budu indirektne ili direktne. U scenarijima kada je jedna jedinka u dometu (bilo u video, bilo u audio) druge jedinke, takve individue uspostavljaju direktne komunikacione kanale. Kao jedan od primera navodi se privlačenje pčela na osnovu plesa pomoću mahanja krilima (eng. waggle dance). U drugom slučaju, gde individue nisu jedna drugoj u dometu, interakcija i komunikacija se najčešće odvijaju tako što individue menjaju okruženje u kojem egzistiraju i na taj način indirektno interaguju sa drugim jedinkama u grupi. Fenomen koji opisuje ovakvo ponašanje naziva se stigmetrija (eng. stigmetry) [71]. Relativno naprednu formu stigmetrije ispoljavaju kolonije mrava, kada krećući se, mravi ostavljaju tragove u vidu supstance koja se naziva feromon i koja omogućava indirektno komuniciranje između članova kolonije promenama okruženja.

Prema [29], osnovni principi na kojima se algoritmi rojeva baziraju su: ne postoji centralna komponenta koja kontroliše i koordinira ponašanje individua, ponašanje grupe proizilazi na osnovu akcija sinhronizovanog i kooperativnog ponašanja jedinki, interakcije između homogenih individua su paralelne i asihnrone, komunikacija između individua se u velikoj meri obavlja putem indirektnih veza u obliku stigmetrije, ciljevi grupe nisu preambiciozni, poput preživaljavanja u neprijateljskom i dinamičnom okruženju, pronalaženje izvora hrane i slično.

Kao još jedna važnija karakteristika algoritama rojeva navodi se sposobnost samoorganizacije. Zahvaljujući samoorganizaciji, grupe jedinki mogu samostalno, bez usmeravačkih i kontrolnih aktivnosti od strane centralne komponente, da ostvare neki cilj, ili da reše određeni problem. Pritom, jedinke donose odluke samo na osnovu lokalno raspoloživih informacije i nemaju uvida u „globalnu sliku". U literaturi se najvčešće navode četiri osnovna principa samoorganizacije [29]: pozitivna i negativna povratna sprega (eng. positive and negative feed-back), višestruke interakcije (eng. multiple interactions) i slučajnost (eng. randomness).

Sa ciljem što kvalitetnijeg i realnijeg modeliranja prirodnih okruženja i sistema, metaheuristike rojeva su usvojile sledeće principe [59]:

- Bliskost (eng. proximity principle) - sposobnost individua u grupi da izvršavaju jednostavna izračunavanja koja zavise od okruženja u kojem se nalaze;

- Kvalitet (eng. quality principle) - zahtevi da roj može da generiše odgovor na kvalitativne faktore;
- Različitosti odgovora (eng. diverse response principle) - distribuiranost resursa u regionu pretrage, tako da je svaki agent pod minimalnim uticajem promena u okruženju i
- Stabilost i prilagođavanje (eng. stability and adaptability principles) - sposobnost da se grupa prilagođava promenama u okruženju bez ishitrenih promena u modelima svog ponašanja, gde svaka promena troši određeni nivo energije.

Jedan od najvažnijih aspekata metaheuristika inteligencije rojeva je način na koji mehanizam koji simulira grupe živih organizama izvršava proces pretrage. Dve suštinske komponente procesa pretrage svakog algoritma roja su eksploatacija (eng. exploitation) i eksploracija (eng. exploration) [160]. Ove komponente su u literaturi poznate i pod nazivima intenzifikacija (eng. intensification) i diversifikacija (eng. diversification).

Mehanizam intenzifikacije, koji se u većini slučajeva izvodi sa visokim stopama konvergencije, predstavlja komponentu koja koristi lokalne informacije u procesu pretrage, sa ciljem da se generišu bolja (kvalitetnija) kandidat rešenja od postojećih. Diverzifikacija se s druge strane odnosi na istraživanje prostora pretrage, da bi se generisala različita rešenja, koja se nalaze dalje od postojećih rešenja [156].

Kao jedan primer mehanizma intenzifikacije može da se navede osnovna jednačina pretrage algoritma kolonije veštačkih pčela (eng. artificial bee colony - ABC), kojom se generiše novo rešenje u susedstvu (okolini) starog (trenutnog) rešenja [52]:

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}), \tag{43}$$

gde v_{ij} i x_{ij} označavaju j-ti parametar novog i starog i-tog rešenja u populaciji, respektivno, ϕ_{ij} je pseudo-slučajan broj iz intervala $[-1, -1]<$ dok x_{kj} označava j-ti parametar slučajnog rešenja k iz populacije.

U nastavku su navedene jednačine kojima se modelira procedura eksploatacije algoritma slepog miša (eng. bat algorithm - BA), koje ažuriraju brzinu i pozicije svakog rešenja u populaciji [155]:

$$f_i = f_{min} + (f_{max} - f_{min})\beta, \tag{44}$$

$$v_i^t = v_i^{t-1} + (x_* - x_i^{t-1})f_i, \tag{45}$$

$$x_i^t = x_i^{t-1} + v_i^t, \tag{46}$$

gde parametri f_i i v_i označavaju vrednost frekvencije i brzine slepog miša i, respektivno, $\beta \in [0, 1]$ je pseudo-slučajan broj generisan uniformnom distribucijom, dok se oznaka x_* odnosi na trenutno najbolje rešenje u populaciji.

Jedan od najvećih izazova, kako u implementiranju metaheuristika rojeva, tako i u primeni, je usklađivanje balansa (kompromisa) između eksploatacije i eksploracije (eng. exploitation-exploration trade-off). Od toga da li je odnos između intenzifikacije i diversifikacije dobro podešen (usklađen), ili nije, zavisi kvalitet rešenja (globalno najbolje rešenje) i brzina konvergencije (prosečni kvalitet rešenja).

U prvom slučaju, kada je balans između ova dva procesa neuravnotežen u smeru ekploatacije, proces pretrage će najverovatnije brzo da konvergira, ali uz smanjenu verovatnoću pronalaska optimalnog regiona prostora pretrage [126]. U ovom slučaju, već u ranijim fazama izvršavanja algoritma, populacija može da izgubi diverzitet (različitost), što može da ima implikacije u vidu preuranjene konvergencije (eng. premature convergence). Na ovaj način, samo u nekim izvršavanjima algoritma, kada algoritam ima „više sreće", pretraga će da konvergira ka optimalnom domenu prostora pretrage.

U drugom slučaju, ako je balans između intenzifikacije i diversifikacije pomeren u korist diversifikacije (eksploracije), algoritam će sa većom verovatnoćom da pronađe domen prostora pretrage gde se nalazi optimum, ali će sporo da konvergira. U ovakvom scenariju, prosečna rešenja će biti bolja, ali će zato najbolja rešenja da budu lošija.

U praktičnim implementacijama algoritama rojeva upotreba samo procesa eksploatacije i eksploracije nije dovoljno i zbog toga je potrebno da se primeni i neki dodatni mehanizam, koji će dodatno da ubrza i da unapredi proces pretrage. jedan od najčešće korišćenih mehanizama je permanentno ažuriranje trenutno najboljeg rešenja (eng.

survival of the fittest), ili primena elitističke selekcije (eng. elitist selection), čime se osigurava da najbolja rešenja neće biti izgubljena i da će biti preneta u sledeću generaciju (iteraciju) [158].

6.2 Prikaz osnovnih implementacija algoritama rojeva koji su unapređeni

U ovom poglavlju prikazani su funkcionalni detalji osnovnih verzija metaheuristika rojeva koji su unapređeni tokom istraživanja koje je sprovedeno za potrebe ove disertacije. Prvo je opisan algoritam optimizacije monarh leptirovima, a zatim pristup optimizacije jatom kitova.

6.2.1 Algoritam optimizacije monarh leptirovima

Algoritam optimizacije monarh leptirovima (eng. monarch butterfly optimization - MBO), predložili su i kreirali 2015 godine, Wang i Deb [143]. U ovom prvom publikovanom naučnom radu, koji opisuje MBO algoritam, prikazani su rezultati testiranja na širem setu standardnih benčmark funkcija bezuslovne optimizacije sa ograničenjima vrednosti promenljivih (eng. bound-constrained benchmarks). Na osnovu rezultata testiranja, pokazano je da ovaj relativno novi algoritam inteligencije rojeva ima odličan potencijal u rešavanju NP teških optimizacionih problema.

Ubrzo nakon što je nastao, MBO algoritam je testiran i na praktičnim NP teškim problemima. Tako na primer, MBO je adaptiran za rešavanje problema planiranja mreže identifikacije pomoću radio frekvencija [108], kao i za problem lokalizacije čvorova u bežičnim senzorskim mrežana (eng. wireless sensor networks - WSN). Takođe, pretragom literature, vidi se da su dostupne i hibridizovane verzije MBO algoritma [119], [117].

Sa ciljem uspešne implementacije MBO algoritma, primenjene su sledeće aproksimacije realnog sistema kojeg MBO simulira [143]:

1. čitava populacija monarh leptirova nalazi se samo na dve lokacije - Teritorija 1 i Teritorija 2;

2. primenom operatora migracije i adaptacije na leptirove iz Teritorije 1 ili Teritorije 2, generiše se potomak monarh leptira;

3. ako je podobnost potomka veća od podobnosti roditelja, potomak se čuva i prenosi se u sledeću generaciju (iteraciju u izvršavanju algoritma). U suprotnom slučaju, potomak se odbacuje iz populacije, dok se roditelj nepromenjen prenosi u sledeću generaciju i

4. na rešenja (leptirove) iz populacije najvećeg kvaliteta (najbolje podobnosti), opeatori se ne primenjuju i takva rešenja se prenose u sledeću iteraciju u trenutnoj formi. Na ovaj način, primenom elitne selekcije, performanse algoritma neće biti degradirane iz jedne iteracije u drugu, već će u najgorem slučaju da ostanu iste.

Kako bi se održala konzistentnost sa terminologijom optimizacionih algoritama, umesto pojma „leptir", u tezi je korišćen termin „rešenje" dok su za termine „Teritorija 1" i „Teritorija 2" korišćeni pojmovi „podpopulacija 1" i „podpopulacija 2", respektivno.

U fazi inicijalizacije MBO algoritma, prvo se određuje broj rešenja u podpopulaciji 1 (N_{sp1}), i podpopulaciji 2 (N_{sp2}). Za ovu svrhu koriste se jednačine (47) i (48) [143].

$$N_{sp1} = ceil(p \cdot N_p), \tag{47}$$

$$N_{sp2} = N_p - N_{sp1}, \tag{48}$$

gde N_p predstavlja broj rešenja u celoj populaciji (broj rešenja u podpopulaciji 1 + podpopulaciji 2), funkcija $ceil(x)$, zaokružuje argument x na najbliži celi broj koji je veći od, ili je jednak x, dok je odnos broja rešenja u podpopulaciji 1 (N_{sp1}) prema ukupnom broju rešenja (N_p), označen kao p.

Parametar p ima veoma važnu ulogu u procesu pretrage MBO algoritma. Podešavanjem vrednosti ovog parametra, uspostavlja se relativni uticaj rešenja u podpopulaciji 1 prema podpopulaciji 2. Ako je vrednost p velika, onda većina rešenja u celoj populaciji biti iz podpopulacije 1, i obrnuto. U originalnoj MBO implementaciji, vrednost parametra p je podešena na 5/12 [143].

Primenom operatora migracije rešenja, novo rešenje u iteraciji $t+1$ se kreira pomoću jedne od sledećih jednačina [143]:

$$x_{i,j}^{t+1} = x_{r_1,j}^{t}, \tag{49}$$

$$x_{i,j}^{t+1} = x_{r_2,j}^{t}, \tag{50}$$

gde $x_{i,j}^{t+1}$ predstavlja j-ti parametar novog x_i rešenja u iteraciji $t+1$, dok $x_{r_1,j}^{t}$ i $x_{r_2,j}^{t}$ označavaju j-tu komponentu od x_{r_1} i x_{r_2} rešenja u trenutnoj iteraciji t, respektivno. Rešenja r_1 i r_2 su pseudo-slučajna rešenja iz podpopulacije 1 i podpopulacije 2, respektivno.

Da li će novo rešenje biti usmereno prema rešenju iz podpopulacije 1 ili podpopulacije 2, zavisi od vrednosti parametra r, koji se računa pomoću sledećeg izraza:

$$r = rand \cdot peri, \tag{51}$$

gde *peri* označava period migracije, koji predstavlja još jedan kontrolni MBO parametar i *rand* označava nasumičan broj izvučen iz uniformisane distribucije. U adaptiranoj verziji MBO algoritma za rešavanje problema raspoređivanja poslova na klaud, kao i u originalnoj MBO implementaciji [143], vrednost parametra *peri* je podešena na 1.2.

Ako je uslov $r \leq p$ zadovoljen, onda se j-ta komponenta rešenja x_i u generaciji $t+1$ generiše na osnovu jednačine (49). U obrnutom slučaju za generisanje j-te komponente rešenja x_i u generaciji $t+1$, koristi se jednačina (50). Operator migracije rešenja se primenjuje samo na rešenja iz podpopulacije 1.

Druga procedura koja izvodi MBO proces pretrage je operator podešavanja rešenja. Ovaj operator izvršava oba procesa, eksploraciju i eksploataciju. Pravac procesa pretrage zavisi od vrednosti generisanog pseudo-slučajnog broja *rnd*, gde mogu da se izdvoje dva slučaja.

U prvom slučaju, ako je uslov $rnd \leq p$ zadovoljen, onda se novo rešenje generiše primenom sledeće jednačine na sve parametre (komponente) trenutnog rešenja [143]:

$$x_{i,j}^{t+1} = x_{best,j}^{t}, \tag{52}$$

gde $x_{i,j}^{t+1}$ predstavlja j-ti parametar novog rešenja i, $x_{best,j}^{t}$ je j-ti parametar trenutno najboljeg rešenja u celoj populaciji (eng. current best solution).

U drugom slučaju, ako je uslov da je $rnd > p$ tačan, onda se novo rešenje u iteraciji $t+1$ dobija primenom jednačine (53) na sve parametre starog rešenja iz

iteracije t:

$$x_{i,j}^{t+1} = x_{r_3,j}^{t}, \tag{53}$$

gde je $x_{r_3,j}^{t}$, j-ti parametar nasumično izabranog rešenja r_3 iz podpulacije 2.

Osim svega navedenog, ukoliko je zadovoljen uslov $rnd > BAR$, novo generisana rešenja se dalje modifikuju primenom sledećeg izraza [143]:

$$x_{i,j}^{t+1} = x_{i,j}^{t+1} + \alpha \times (dx_j - 0.5), \tag{54}$$

gde BAR predstavlja brzinu podešavanja (prilagođvanja) kandidat rešenja, dok dx označava dužinu koraka pretrage (eng. walk step), koji se izračunava na osnovu Lévy letova (eng. Lévy flights) [143]:

$$dx = Levy(x_i^t). \tag{55}$$

Operator podešavanja rešenja (eng. butterfly adjusting rate - BAR) se primenjuje samo na rešenja iz podpopulacije 2.

Faktor težine α, koji se koristi u jednačini 54 se izračunava kao [143]:

$$\alpha = S_{max}/t^2, \tag{56}$$

gde je S_{max} maksimalna dužina koraka u kome individua (rešenje) može da se kreće u jednom vremenskom trenutku izvršavanja algoritma, dok t označava trenutku iteraciju.

Vrednost parametra α reguliše balans između eksploatacije i eksploracije. Kada je vrednost α veća, korak pretrage je duži i proces eksploracije ima veći uticaj na ceo proces pretrage. S druge strane, sa smanjenjem vrednosti α, uticaj procesa intenzifikacije na pretragu se povećava.

Pseudo-kod osnovne MBO implementacije prikazan je u Algoritmu 4.

Algoritam 4 Pseudo-kod osnovnog MBO algoritma

Inicijalizacija početne populacije i kontrolnih parametara
Evaluacija podobnosti. Evaluacija svakog rešenja u populaciji u odnosu na funkciju cilja i računanje podobnosti.
while $t < MaxGen$ **do**
  Sortiranje svih rešenja u populaciji na osnovu podobnosti.
  Podela početne populacije na podpopulaciju 1 i podpopulaciju 2.
  for $i = 1$ **to** NP_1 (sva rešenja u podpopulaciji 1) **do**
    Generisanje novih rešenja (individua) primenom operatora migracije.
  end for
  for $j = 1$ **to** NP_2 (sva rešenja u podpopulaciji 2) **do**
    Generisanje novih rešenja (individua) primenom operatora prilagođavanja (adaptacije).
  end for
  Spajanje novo kreirane podpopulacije 1 sa novo kreiranom podpopulacijom 2.
  Evaluacija nove populacije.
  Inkrement brojača trenutne iteracije t.
end while
return Najbolje rešenje iz populacije.

6.2.2 Algoritam optimizacije jatom kitova

Algoritam optimizacije jatom kitova (eng. whale optimization algorithm - WOA) je prvi put predložen 2016. godine za rešavanje problema globalne optimizacije bez ograničenja i problema kontinulane optimizacije sa ograničenjima, gde je testiran na dobro poznatim inženjerskim instancama benčmark problema [80]. Tvorci ovog pristupa su od Mirjalili i Lewis. Metaheuristika WOA je inspirisana sa strategijom lova u obliku mreže mehurića (eng. bubble-net hunting strategy) grbavog kita (eng. humpback whale).

Samo nekoliko godina nakon što je prvi put implementiran, WOA algoritam se pozicionirao kao robustni optimizator sposoban za rešavanje različitih vrsta NP teških problema, posebno u modifikovanim/unapređenim implementacijama [70], [67].

Proces pretrage WOA simulira strategiju lova u obliku mreže mehurića grbavog kita. U prirodi, dok primenjuju ovakvu strategiju, grbavi kitovi pokazuju visok nivo kooperativnog i sinhronizovanog ponašanja. Lov u obliku mreže mehurića se izvodi tako što se kitovi prvo pozicioniraju ispod potencijalnog plena, a zatim simultano kreiranju mehuriće i podižu se, sledeći kružnu putanju, ka površini vode. Na ovaj način, mehurići stimulišu potencijalni plen da se kreće prema površini vode, gde

postaju lak „zalogaj" za grbavog kita [41], [80].

Kao i kod svakog algoritma inteligencije rojeva, dve osnovne komponente procesa pretrage metaheuristike WOA su globalna (eksploracije) i lokalna (eksploatacije) pretraga. Procesom eksploatacije se modelira kruženje grbavog kita oko plena i spiralna (kružnu) strategija napada sa kružnom mrežom mehurića, dok se procesom eksploracije modelira pseudo-slučajana potraga plena.

Obzirom da WOA priprada kategoriji populaciono optimizacionih metoda, individue u populaciji izvršavaju proces pretrage nezavisno, dok u isto vreme uspostavljaju formu indirektne komunikacije i stigmetrije, koja poboljšava proces pretrage i omogućava metaheuristici da brže konvergira ka optimumu. Prema osnovnoj WOA terminologiji, trenutno najbolje (globalno najbolje) kandidat rešenje (rešenje sa najboljom vrednošću funkcije podobnosti), predstavlja ciljni plen (eng. target prey), dok svako drugo rešenje predstavlja grbavog kita (individua) u populaciji.

Slično kao i u slučaju MBO metaheuristike, s ciljem uspostavljanja konzistentnosti sa osnovnom terminologijom koja se koristi u domenu algoritama rojeva, i kako bi se izbegle nepravilnosti, izraz kit i plen u nastavku nisu korišćeni, i umesto ovih izraza su upotrebljeni izrazi kandidat rešenja i trenutno najbolje rešenje.

Tokom faze eksploatacije (intenzifikacije), svako kandidat rešenje izvršava pretragu u svom bliskom okruženju i usmereno je prema poziciji sa trenutno najboljim globalnim rešenjem. Nakon izračunavanja podobnosti svakog rešenja u populaciji, pozicije svih rešenja u populaciji se ažuriraju u odnosu na poziciju globalno najboljeg rešenja (individue koja ima najveću vrednost funkcije podobnosti) [80]:

$$\vec{D} = |\vec{C} \cdot \vec{X}^*(t) - \vec{X}(t)| \tag{57}$$

$$\vec{X}(t+1) = \vec{X}^*(t) - \vec{A} \cdot \vec{D}, \tag{58}$$

gde $\vec{X}(t)$ i $\vec{X}^*(t)$ označavaju kandidat rešenja i trenutno najbolje rešenje u iteraciji t, respektivno, $\vec{A}$ i $\vec{C}$ predstavljaju vektore koeficijenata pretrage, dok se simbol $\cdot$ odnosi na operator unakrsnog množenja.

Sledeći izraz se koristi za računanje vektora koeficijenata $\vec{A}$ i $\vec{C}$ [80]:

$$\vec{A} = 2\vec{a} \cdot \vec{r} - \vec{a} \tag{59}$$

$$\vec{C} = 2 \cdot \vec{r} \tag{60}$$

U jednačinama (59) i (60), $\vec{r}$ označava uniformno distribuirani pseudo-slučajni vektor u intervalu $[0, 1]$, dok je $\vec{a}$ vektor koji se u osnovnoj verziji WOA implementacije linearno smanjuje od 2 do 0 tokom izvršavanja algoritma [80].

Na osnovu jednačina (57) - (60), vidi se da kandidat rešenje $\vec{X}$ u svakoj iteraciji može da se usmeri prema bilo kojoj poziciji iz okruženja trenutno najboljeg rešenja $\vec{X}^*$, podešavanjem vrednosti vektora koeficijenata $\vec{A}$ i $\vec{C}$ uz upotrebu pseudo-slučajanog vektora $\vec{r}$.

Mehanizam smanjivanja poluprečnika kružnih putanja (eng. the shrinking encircling mechanism) i putanja spiralnog oblika (eng. spiral-shaped path), su osnovni matematički modeli koji oponašaju strategiju napada u obliku mreže mehurića.

Izvršavanje mehanizma smanjivanja poluprečnika kružnih putanja modelira se linearnim smanjivanjem vrednosti vektora $\vec{a}$, od vrednosti 2 do 0 (u osnovnoj WOA implementaciji), tako što se u svakoj iteraciji izvršavanja algoritma koristi sledeći izraz [80]:

$$\vec{a} = 2 - t\frac{2}{maxIter}, \tag{61}$$

gde t i $maxIter$ označavaju trenutni i maksimalni broj iteracija u jednom izvršavanju algoritma, respektivno.

Autori rada u kojem je originalna verzija WOA metaheuristike prvi put prikazana [80] preporučuju da vrednost $\vec{A}$ treba da se podesi u intervalu $[-1, 1]$. Na ovaj način, ažurirana pozicija trenutnog rešenja može da se nalazi bilo gde između njegove trenutne pozicije i pozicije globalno najboljeg rešenja iz populacije.

Drugi mehanizam, koji vodi proces eksploatacije, putanja spiralnog oblika, izvršava se u dva koraka: prvo se izračunava razdaljina između globalno najboljeg rešenja $(\vec{X}^*(t))$ i trenutnog rešenja $(\vec{X}(t))$ u iteraciji t, a zatim, nova (ažurirana) pozicija kandidat rešenja $(\vec{X}(t+1))$, se određuje primenom tzv. spiralne jednačine [80]:

$$\vec{X}(t+1) = \vec{D'} \cdot e^{bl} \cdot \cos(2\pi l) + \vec{X}^*(t), \tag{62}$$

gde $\vec{D'}$, kao razdaljina između i-tog kandidat rešenja i globalno najboljeg rešenja, može da se izrazi kao $\vec{D'} = |\vec{X}^*(t) - \vec{X}(t)|$, gde b predstavlja konstantu koja definiše

oblik logaritamske spirale, dok l označava pseudo-slučajan broj u opsegu od -1 do 1.

U prirodi se grbavi kitovi kreću oko plena tako što kreiraju putanju spiralnog oblika, dok u isto vreme simultano smanjuju poluprečnik kružnih putanja kojima se kreću oko plena. Ovakvo ponašanje se emulira tako što se u svakoj iteraciji izvršavanja algoritma, mehanizmi smanjivanja poluprečnika kružnih putanja i putanja spiralnog oblika primenjuju sa podjednakom verovatnoćom p:

$$\vec{X}(t+1) = \begin{cases} \vec{X}^*(t) - \vec{A} \cdot \vec{D} & \text{, if } p < 0.5 \\ \vec{D'} \cdot e^{bl} \cdot \cos(2\pi l) + \vec{X}^*(t) & \text{, if } p \geq 0.5 \end{cases} \tag{63}$$

Za svako rešenje i u svakoj iteraciji, parametar p se bira na osnovu nasumične distribucije između 0 i 1.

Druga komponenta procesa pretrage, eksploracija (diversifikacija), izvodi se tako što se svako kandidat rešenje iz populacije ažurira u odnosu na poziciju nasumično odabranog rešenja, umesto u odnosu na poziciju globalno najboljeg rešenja, kao što je slučaj tokom procesa eksploatacije.

U fazi eksploracije koristi se vrednost vektora koeficijenta $\vec{A}$. Ako su vrednosti komponenti vektora $\vec{A}$ veće od ili jednake od 1 ($|A| \geq 1$), ažurirane pozicije kandidat rešenja će biti usmerene prema nasumično odabranim rešenjima i na taj način se izvršava globalna pretraga.

Sledećim izrazom se modelira faza eksploracije WOA metaheuristike [80]:

$$\vec{X}(t+1) = \vec{X}_{rnd}(t) - \vec{A} \cdot \vec{D}, \tag{64}$$

gde se $\vec{D}$, kao razdaljina između i-tog kandidat rešenja i nasumičnog rešenja rnd iz populacije u iteraciji t, izračunava kao $\vec{D} = |\vec{C} \cdot \vec{X}_{rnd}(t) - \vec{X}(t)|$.

Pseudo-kod originalne WOA implementacije, koji prikazuje osnovne korake u izvršavanju algoritma, dat je u Algoritmu 5.

Algoritam 5 Pseudo-kod WOA metaheuristike

Inicijalizacija. Generisanje početne slučajne populacije $X_i (i = 1, 2, 3, ..., N)$ i inicijalizacija vrednosti kontrolnih parametara.
Izračunavanje podobnosti. Izračunavanje funkcije podobnosti svakog kandidat rešenja i određivanje globalno najboljeg rešenja X^*
while $(t < maxIter)$ **do**
  for svako kandidat rešenje **do**
    Ažuriranje vrednosti parametara A, C, a, l i p
    if $p < 0.5$ **then**
      if |A|<1 **then**
        Ažuriranje pozicija kandidat rešenja X korišćenjem jednačine (58)
      else
        Biranje slučajnog rešenja rnd iz populacije.
        Ažuriranje pozicija kandidat rešenja X korišćenjem jednačine (64)
      end if
    else
      Ažuriranje pozicija kandidat rešenja X primenom jednačine (62)
    end if
  end for
  Ako su vrednosti bilo kog parametara bilo kojeg rešenja izvan granica izvodljivog regiona prostora pretrage, postaviti vrednosti takvih parametara na donje ili gornje granice izvodljivog domena prostora pretrage.
  Evaluacija svih rešenja u populaciji i računanje podobnosti.
  Ažuriranje pozicija globalno najboljeg rešenja X^* ako je potrebno.
  $t = t + 1$
end while
return Najbolje rešenje (X^*) iz populacije

6.3 Pregled literature

U ovom delu disertacije prikazane su primene poznatijih algoritama metaheuristika rojeva za rešavanje NP teških problema. Poseban fokus stavljen je na adaptacije algoritama rojeva za rešavanje problema raspoređivanja poslova i balansiranja opterećenja u klaud računarstvu. Obzirom na veliki broj implementacija, na početku ovog poglavlja ukratko su prikazane i neke od aplikacija GA.

Pregledom literature ustanovljeno je da je metaheuristika GA robustan i efikasni optimizator, koja se već dugi niz godina uspešno primenjuje na širok spektar NP teških problema [25], [90], [6]. Poznate su i adaptacije algoritma GA na izazove iz domena klaud računarstva. Tako na primer, u [35] prikazano je rešavanje hibridnog modela klaud računarstva i Interneta inteligentnih uređaja (eng. internet of things - IoT) za upravljanje velikim skupovima podataka u klaud aplikacijama koje se primenjuju u zdravstvu. Implementacija jedne unapređene verzije GA za rešavanje izazova raspoređivanja poslova na klaudu pomoću prioritetnih redova data je u [58], dok je u [147] izazov balansiranja opterećenja u klaud okruženju uspešno rešavan primenom ove metaheuristike.

Metaheuristike inteligencije rojeva, kao populacioni, stohastički i iterativni pristupi, koji pokušavaju da poboljšaju potencijalno rešenje problema u unapred definisanom broju ciklusa (iteracija), demonstrirale su superiorne performanse u rešavanju brojnih problema globalne i/ili kombinatorne optimizacije. Bez obzira što su u literaturi zastupljene brojne implementacije algoritama rojeva za rešavanje problema iz domena klaud računarstva [96], [17], [82], obzirom na heterogenost ove familije algoritama, kao i na potencijal koji imaju u domenu rešavanja NP teških izazova, u ovom domenu postoji još mnogo mogućnosti za istraživanje.

Jedan od najpoznatijih predstavnika inteligencije rojeva, optimizacija rojevima čestica (eng. particle swarm optimization - PSO) [57], [20], inspirisana je kretanjem jata ptica u prirodi. Istraživači Eberhart i Kennedy su jedni od pionira u ovoj oblasti, koji su još početkom 90tih godina prošlog veka, uspeli da efikasno modeliraju i inkorporiraju pravila kretanja jata ptica u optimizacioni algoritam. Mehanizam pretrage PSO algoritma izvodi se u odnosu na najbolje pronađeno rešenje i njegovog suseda.

Algoritam PSO je modifikovan i unapređen za rešavanje mnogih praktičnih

problema [84], [164], [101]. Takođe, pretragom literature vidi se da je PSO uspešno primenjivan i za probleme iz domena klaud računarstva. U [35], autori su predložili novi model sa ciljem optimizacije izbora virtuelnih mašina u klaud IoT aplikacijama zdravstvenih servisa i adaptirali originalni PSO algoritam i paralelni PSO (PPSO) za rešavanje ovog izazova na klaud infrastrukturi. Osim ove implemetacije, u [63] prikazana je implementacija PSO algoritma za problem raspoređivanja poslova na klaudu sa ciljevima minimizacije potrošene energije i troškova, uz ograničenje rokova izvršavanja zadataka.

Još jedan poznatiji predstavnik metaheuristika inteligencije rojeva je algoritam kolonije veštačkih pčela (eng. artificial bee colony - ABC), koji simulira kooperativno ponašanje pčela u rojevima. Algoritam ABC je prvi predložio Karaboga za probleme neprekidne optimizacije bez ograničenja [50], da bi ga nakon toga Karaboga i Bastuk adaptirali i za rešavanje problema globalne optimizacije sa ograničenjima [51], [52]. Ova metaheuristika pretragu izvodi pomoću tri vrste veštačkih agenata: pčela radilica (eng. employee bee), pčela posmatrača (eng. onlooker bee) i izviđača (eng. scout bee). ABC pristup je uspešno testiran na širem setu benčmark funckija [12], kao i na praktičnim NP teškim problemima, poput optimizacije finansijskog portfolija [135], [138], planiranja RFID mreže [8], [137], inženjerskih problema [140] i WSN lokalizacije [26]. Za rešavanje problema u klaud računarstvu, ABC je implementiran u [163], sa ciljem uspostavljanja sistema efikasnog upravljanja heterogenim klaud resursima. U [53], autori su prikazali razvijeni hibridni ABC algoritam, koji je značajno uspeo da smanji potrošnju energije tokom migracije virtuelnih mašina.

Metaheuristika optimizacije pretragom svitaca (eng. firefly algorithm - FA), takođe je značajan predstavnik metaheuristika inteligencije rojeva. Tvorac ovog algoritma je Xin-She Yang [152], koji je, inspirisan svetlosnom signalizacijom i ponašanjem jata svitaca, kreirao algoritam FA i adaptirao ga za optimizaciju neprekidnih problema bez ograničenja. U svetu svitaca, svetlost koja treperi može da se sagledava u kontekstu privlačenja, ili kao mehanizam za upozoravanje od napada. Privlačenje svitaca i mehanizmi upozoravanja bazirani na svetlosnoj signalizaciji, modelirani su i inkorporirani u osnovne jednačine pretrage FA metaheuristike [154].

Istraživanjem dostupnih izvora literature može da se vidi da je FA algoritam uspešno prilagođen za heterogene praktične NP teške probleme [14], [13], [139].

Algoritam je uspešno testiran na standardnim bančmark problemima u modifikovanim [112] i hibridizovanim verzijama [141]. Kao jedna od novijih adaptacija navodi se kombinacija FA metaheuristike sa konvolutivnim neuronskim mrežama (eng. convolutional neural networks - CNNs), gde je FA pokazao da uspešno može da optimizuje vrednosti hiper-parametara CNN mreže [111]. FA algoritam je takođe implementiran za rešavanje problema u klaud računarstvu. Tako na primer, primenom FA pristupa prikazanog u [54], značajno je unapređen mehanizam balansiranja opterećenja u klaud okruženju. Kao još jedan primer navodi se implementacija FA metaheuristike za raspoređivanje poslova koji zavise jedni od drugih (eng. workflow scheduling), gde je takođe postigao značajne rezultate [123].

Algoritam optimizacije pretragom kukavica (eng. cuckoo search - CS), koji je inspirisan inteligentnim ponašanjem jata kukavica, kreirali su Yang i Deb [159]. Sličan algoritam, metaheuristika pretrage slepog miša (eng. bat algorithm - BA), koja modelira fenomen eholokacije slepih miševa, takođe je modelirao i predložio Yang [155]. Algoritmi su uspešno testirani na standardnim skupovima benčmark funkcija [9], [161]. Obe metaheuristike imaju široku primenu u rešavanju praktičnih NP teških problema, kao su problemi planiranja RFID mreže [136], treniranja veštačke neuronske mreže (eng. artificial neural network - ANN) [134] i portfolio optimizacije [105]. Implementacije metaheuristika CS i BA za probleme u klaud računarstvu koji mogu da se pronađu u dostupnoj literaturi, obuhvataju probleme raspoređivanja radnog toka [55], [94], kompozicije klaud usluga [150], raspoređivanja poslova [4] i balansiranja opterećenja [151].

Algoritam vatrometa (eng. fireworks algorithm - FWA) spada u poznatije metaheuristike koje su predložili Tan i Zhu [125]. Prikaz FWA metaheuristike, inspirisane eksplozijom varnica koje nastaju puštanjem vatrometa, prvi put je objavljen 2010. godine. Od svog nastanka, FWA je primenjen za rešavanje brojnih praktičnih NP teških problema, među kojima su problemi obrade digitalnih slika [131], [132], planiranja RFID mreže [133], [106], kao i brojni drugi teški optimizacioni izazovi [130], [14]. U literaturi je takođe pronađena i jedna implementacija metaheuristike FWA za problem raspoređivanja poslova na klaudu, gde je ovaj algoritam pokazao da može da postigne bolju vrednost makespan indikatora od drugih sličnih algoritama [66].

Algoritam pretrage moljaca (eng. moth search - MS) spada u relativno nove

metaheuristike inteligencije rojeva. Mehanizam pretrage MS algoritma simulira privlačenje ka izvorima svetlosti i Lévy letove moljaca. Ovaj pristup je kreirao Wang 2018. godine [142]. Algoritam MS je testiran na benčmark funkcijama sa ograničenjima i bez ograničenja [107], [142], ali i na razne praktične probleme, kao što su izazovi u domenu bežičnih senzorskih mreža za rešavanje problema lokalizacije [109] i problem pozicioniranja dronova [118]. U literaturi iz oblasti računarstva pronađen je jedan rad u kome je hibridizovani MS algoritam adaptiran i implementiran za problem raspoređivanja poslova u klaud okruženju [34].

Kao još jedan relativno novi predstavnik metaheuristika inteligencije rojeva, kojeg je takođe kreirao Wang, navodi se algoritam optimizacije krdom slonova (eng. elephant herding optimization - EHO). Prva implementacija algoritma EHO testirana je za rešavanje problema globalne optimizacije [145], [144]. Metaheuristika EHO simulira socijalne interakcije između jedinki krda slonova pod vođstvom i uticajem ženke matrijarhata, gde kada odrastu, mužijaci napuštaju klan kako bi živeli nezavisno, ali još uvek komuniciraju sa ostatkom krda. Zbog obećavajućih rezultata koje EHO postiže u testiranjima na standardnim benčmark funkcijama, ovaj algoritam je ubrzo nakon nastanka adaptiran i implementiran i za praktične NP teške probleme, među kojima se istuču problem lokalizacije u bežičnim senzorskim mrežama [115], [116] i izazov statičkog raspoređivanja dronova [116]. Takođe, u literaturi su dostupne i hibridne verzije EHO metaheuristike [113]. Istraživanjem raspoloživih izvora, zaključeno je da ne postoje primene EHO algoritma na rešavanje problema iz domena klaud računarstva.

Algoritam rasta drveća (eng. tree growth algorithm - TGA) nastao je sa ciljem simuliranja prirodne konkurencije među drvećem tokom apsorpcije hrane iz zemlje. Algoritam TGA su kreirali i predložili Armin Cheraghalipour i Mostafa Hajiaghaei-Keshteli 2017 godine [27]. Metaheuristika TGA ima mnogobrojne primene za probleme globalne optimizacije [120], kao i praktične probleme u domenu lokalizacije u bežičnim senzorskim mrežama [116]. Godine 2019., TGA je takođe primenjen za probleme raspoređivanja poslova u klaud računarstvu, u originalnoj i unapređenoj verziji [110]. Originalni TGA algoritam je poboljšan uvođenjem dinamičkog prilagođavanja parametara pretrage intenzifikacije i diverzifikacije.

Osim algoritama inteligencije rojeva koji su navedeni u ovom poglavlju, postoje i

drugi predstavnici koji su ostvarili zapažene performanse u rešavanju heterogenih praktičnih problema. Jedan od najstarijih predstavnika ove familije prirodom-inspirisanih metaheuristika je algoritam optimizacije mravljih kolonija (eng. ant colony optimization - ACO) [71], koji se najčešće primenjuje na rešavanje grafovskih problema. Metaheuristika ACO simulira indirektnu komunikaciju između mrava u koloniji, gde mravi ostvaruju interakcije ispuštanjem supstance koja se naziva feromon. Primer jedne implementacije ACO algoritma može se videti u [45]. Algoritam koji se bazira na procesu donošenja odluka grupe ljudi (eng. brain storm optimization - BSO) [100] takođe ostvaruje značajne rezultate iz različitih domena, kao na primer klasterovanja [128] i planiranja putanje robota [129]. Kao još jedan predstavnik metaheuristika rojeva navodi se algoritam optimizacije jatom planktona (eng. krill herd - KH), koji je postigao značajne rezultate [15], [146].

Za detaljnije informacije o primeni hibridnih metaheuristika rojeva za rešavanje problema raspoređivanja poslova i balansiranja opterećenja na klaudu pogledati [5], [36], dok se sveobuhvatan pregled implementacija algoritama rojeva za ove izazove može naći u [49], [82].

7 Primena hibridnog MBO algoritma na rešavanje problema raspoređivanja poslova i balansiranja opterećenja u klaud računarstvu

Prvi algoritam koji je implementiran i adaptiran za potrebe istraživanja ove doktroske disertacije predstavlja unapređenu verziju osnovne MBO metaheuristike. Teorijskom analizom i praktičnim simulacijama sa osnovnim MBO pristupom, ustanovljeni su određeni nedostaci originalne MBO implementacije. Unapređena verzija MBO metaheuristike eliminiše nedostatke osnovne verzije primenom pristupa hibridizacije sa drugom metaheuristikom. Unapređena MBO verzija testirana je prvo na širem skupu standardnih benčmark problema globalne optimziacije, a zatim i na problemu raspoređivanja poslova i balansiranja opterećenja na klaudu.

Detaljan prikaz osnovne MBO metaheuristike dat je u Poglavlju 6.2.1.

U prvom delu ove glave prikazani su nedostaci originalnog MBO algoritma inteligencije rojeva, gde su pojedinačno istaknuti i objašnjeni svi elementi koji se razmatraju za dalja unapređenja. Nakon toga, polazeći od uočenih nedostataka, u drugom delu ove glave prikazani su implementacioni detalji hibridnog MBO algoritma koji eliminiše nedostatke osnovne verzije.

Sledeći uobičajenu praksu eksperata iz domena metaheuristika rojeva, kada se kreira novi algoritam, ili kada se unapredi verzija postojećeg algoritma, sa ciljem evaluiranja performansi novog pristupa, prvo se izvode testovi na širem skupu benčmark problema. Zbog toga su u trećem delu ove glave prikazani rezultati simulacije na standardnim problemima globalne optimizacije. Takođe, prikazana je i detaljna komparativna analiza unapređenog MBO algoritma sa osnovnim MBO, kao i sa drugim vrhunskim algoritmima inteligencije rojeva, čiji su rezultati publikovani u svetskoj literaturi.

U četvrtom delu, prikazani su rezultati testiranja kreiranog unapređenog MBO algoritma inteligencije rojeva za praktičan problem raspoređivanja poslova u klaud okruženju, korišćenjem dva veštačka skupa podataka (generisanih u okruženju CloudSim simulatora) i realnog skupa podataka, koji su preuzeti iz jedne svetske baze. U oba slučaja, unapređeni MBO testiran je na modelu raspoređivanja poslova u klaud okruženju sa jednom funkcijom cilja (pogledati Poglavlje 4.2). Takođe, u ovom delu

disertacije prikazano je prilagođavanje (adaptiranje) unapređene metaheuristike za problem raspoređivanja poslova na klaudu, kao i detaljna analiza i komarativna analiza sa osnovnom MBO metaheuristikom i drugim heuristikama i metaheuristikama koje su testirane za instancu ovog problema.

Konačno, na kraju ove glave dat je kratak zaključak eksperimenata sa hibridnim MBO algoritmom i izvedene su implikacije na osnovne hipoteze koje su formulisane u Glavi 1.

Rezultati do kojih se došlo ovim istraživanjem publikovani su u međunarodnom časopisu od nacionalnog značaja referisanog u Scopus bazi [121].

7.1 Nedostaci osnovnog MBO algoritma

Izvršavanjem praktičnih simulcija na standardnim benčmark funkcijama bezuslovne optimizacije sa ograničenjima vrednosti parametara, primećeni su određeni nedostaci u originalnoj implementaciji MBO metaheuristike.

Prvi nedostatak se odnosi na nedovoljnu snagu procesa eksploracije. U osnovnoj verziji MBO, operator migracije rešenja (eng. solutions migration operator) izvršava proces eksploatacije tako što pomera rešenja u pravcu postojećih rešenja iz podpopulaciji 1 i podpopulaciji 2. Drugi operator, operator podešavanja rešenja (eng. solutions adjusting operator), izvršava dva procesa: eksploataciju i ekksploraciju. Kada se primeni ovaj operator, eksploatacija se izvršava tako što pomera rešenja u pravcu trenutno najboljeg rešenja u celoj populaciji, ili u pravcu nasumično izabranih rešenja iz podpopulacije 2 (pogledati jednačine (52) i (53)).

Proces eksploracije se izvršava samo u slučaju kada je uslov $rnd > BAR$ zadovoljen. Međutim, čak i u ovom slučaju, intenzitet eksploracije zavisi od faktora težine (eng. weighting factor) α i najveće dužine koraka (eng. maximum walk step) (S_{max}), koja se dinamično smanjuje u toku izvršavanja algoritma (pogledati jednačinu (56)).

Uočeni nedostatak u snazi eksploracije u ranim fazama iteracije izvršavanja algoritma ima veliki uticaj. U slučaju testiranja sa većinom instanci benčmark prblema, zbog nedovoljne snage eksploracije, MBO u ranim fazama izvršavanja, nije mogao da nađe pravi deo prostora pretrage, što je dovelo do zaglavljivanja u nekom u suboptimalnih delova prostora pretrage, i na kraju do loših prosečnih vrednosti. S druge strane, u nekim izvršavanjima, kada algoritam ima više „sreće", prosečne

vrednosti su bolje.

Drugi nedostatak originalne MBO implementacije odnosi se na loše uspostavljen balans između procesa eksploatacije i eksploracije (eng. exploitation-exploration trade-off), koji je postavljen u korist eksploatacije. Podešavanje balansa između ova dva procesa je važna komponenta performansi svakog algoritma inteligencije rojeva.

U ranim fazama izvršavanja algoritma, sa pretpostavkom da optimalan deo domena pretrage nije pronađen, balans između intenzifikacije i diversifikacije bi trebao da naginje ka diversifikaciji. Međutim, u kasnijim fazama, kada je obećavajući domen prostora pretrage pogođen, balans bi da se podesi u korist intenzifikacije, čime bi mogla da se realizuje fina pretrage optimalnog domena prostora pretrage. Testiranjem osnovnog MBO pristupa na standardnim benčmark funkcijama, primećeno je da čak i slučaju kada su vrednosti parametra S_{max} velike, zadovoljavajući balans između eksploracije i eksploatacije nije mogao da se uspostavi.

7.2 Prikaz hibridizovanog MBO algoritma

Sa ciljem da se prevaziđu uočeni nedostaci i mane osnovnog MBO pristupa, kreirana je hibridna MBO metaheuristika. Hibridizacija, kao metoda koja kombinuje snage dve ili više metaheuristika (algoritama) i u isto vreme eliminiše slabosti, dokazana je kao efikasan put ka unapređenju originalnih implementacija metaheuristike rojeva. Neki od primera hibridizacije MBO verzije algoritama se mogu videti iz [119], [117].

Da bi se mehanizam procesa eksploracije unapredio, usvojen je i adaptiran mehanizam odbacivanja istrošenih rešenja (eng. mechanism of discarding exhausted solutions) iz ABC metaheuristike. Za svako rešenje u populaciji, uključen je dodatni atribut *trial*. U svakoj iteraciji, u kojoj trenutno rešenje ne može da se poboljša, parametar *trial* se povećava za jedan. Kada vrednost *trial* parametra za određeno vrešenje dostigne unapred predefinisanu vrednost, za takvo rešenje se kaže da je istrošeno i kao takvo se izbacuje iz populacije. Predefinisana vrednost, koja predstavlja prag za izbacivanje rešenja iz populacije određuje se novim kontrolim parametrom pod nazivom vrednost istrošenosti (eng. exhaustivness value) i ovaj parametar se označava sa *exh*. Izbačeno rešenje iz populacije se zamenjuje slučajnim rešenjem, čime se poboljšava snaga eksploracije [121]:

$$x_{i,j} = \phi \cdot (ub_j - lb_j) + lb_j, \tag{65}$$

gde ub_j i lb_j označavaju donju i gornju granicu vrednosti parametra j, respektivno, dok je ϕ pseudo-slučajan broj iz opsega $[0, 1]$. Ova jednačina se takođe koristi u fazi inicijalizacije algoritma, kada se generiše slučajna početna populacija.

Na osnovu sprovedenih praktičnih simulacija, zaključeno je da ako se mehanizam odbacivanja istrošenih rešenja primeni tokom svih iteracija izvršavanja algoritma, tada, u kasnijim iteracijama, sa pretpostavkom da je proces pretrage konvergirao kao domenu gde se nalazi optimum, može da dođe do scenarija da se previše kvalitetnih rešenja izbaci iz populacije.

Da bi se ovaj problem prevazišao, uveden je drugi kontrolni parametar, okidač za pokretanje mehanizma odbacivanja istrošenih rešenja (eng. discarding mechanism trigger), dmt, koji kontroliše koliko iteracija, od početka izvršavanja algoritma, mehanizam odbacivanja istrošenih rešenja iz populacije će da se izvršava. U većini testova, optimalna vrednost ovog parametra izračunava se primenom izraza: $round(maxIter/1.5)$, gde $maxIter$ označava maksimalni broj iteracija u jednom puštanju algoritma i $round(x)$ je funkcija koja zaokružuje argument x na najbližu celu vrednost. Nakon $round(maxIter/1.5)$ iteracija, mehanizam za odbacivanje istrošenih rešenja se neće izvršavati.

Takođe je primećeno da proces pretrage originalne MBO implementacije u ranim iteracijama ponekada konvergira previše brzo ka suboptimalnom regionu domena pretrage, i da se zbog toga algoritam često zaglavljuje u nekom od lokalnih optimuma. Ovakvo ponašanje je u literaturi poznato pod nazivom preuranjena konvergencija (eng. premature convergence). Ovo je posledica neadekvatno postavljenog balansa između intenzifikacije i diverzifikacije, što je posebno izraženo u mehanizmu pretrage enkapsuliranog u okviru operatora migracije rešenja. Ovaj operator se primenjuje na svaku komponentu rešenja [121].

Da bi se predupredila preuranjena konvergencija, uveden je drugi kontrolni parametar iz ABC metaheuristike, koji se naziva stopa modifikacije (eng. modification rate) i koji se označava sa MR. U metodi koja modelira operator migracije rešenja, za svaku komponentu svakog rešenja iz podpopulacije 1, generiše se pseudo-slučajan broj θ, i samo ako e uslov da je $\theta \leq MR$ zadovoljen, tada će komponente trenutnih

rešenja biti modifikovane prema jednačini (49) ili (50). U suprotnom, ako uslov $\theta \leq MR$ komponente postojećih rešenja neće biti ažurirane [121].

Inkorporiranjem tri dodatna kontrolna parametra i pojam istrošenih rešenja iz ABC metaheuristike, pojačana je snaga eksploracije originalnog MBO pristupa i uspostavljen je bolji balans između intenzifikacije i diverzifikacije.

S obzirom da kreirani hibridni pristup nedostatke osnovne MBO metaheuristike eliminiše uvođenjem mehanizama ABC algoritma, hibridizovani algoritam nazvan je MBO-ABC [121].

Pseudo-kod MBO-ABC metaheuristike je dat u Algoritmu 6.

Algoritam 6 Pseudo kod MBO-ABC metaheuristike

Inicijalizacija. Podešavanje brojača trenutne iteracije t na 1, maksimalnog broja iteracija u jednom izvršavanju $maxIter$, vrednost za p, $peri$,S_{max}, BAR, exh i dmt kontrolne parametre; Generisanje slučajne početne populacije veličine N_p primenom jednačine (65); Određivanje N_{sp1} i N_{sp2} veličine podpopulacije 1 i podpopulacije 2 jednačinama (47) i (48); Inicijalizacija $trial$ za sva rešenja u populaciji na 0.
Evaluacija podobnosti. Izračunavanje podobnosti svih rešenja u populaciji.
while $t < maxIter$ **do**
  Sortiranje svih rešenja u populaciji na osnovu kriterijuma podbnosti.
  Podela početne populacije na podpopulaciju 1 veličine N_{sp1} i podpopulaciju 2 veličine N_{sp2}
  for all rešenja u podpopulaciji 1 **do**
    for all komponente rešenja **do**
      Generisanje pseudo-slučajanog broja θ
      if $\theta \leq MR$ **then**
        Ažuriranje pozicije rešenja primenom jednačine (49) ili jednačine (50)
      end if
    end for
    Izbor između starog i novog rešenja u odnosu na podobnost.
    Ako je staro rešenje izabrano, povećati vrednost $trial$ parametra
  end for
  for all rešenja u podpopulaciji 2 **do**
    for all komponente rešenja **do**
      Ažuriranje pozicije rešenja primenom jednačina (52)–(55)
    end for
    Izbor između starog i novog rešenja u odnosu na podobnost.
    Ako je izabrano staro rešenje, uvećati vrednost $trial$ parametra
  end for
  for all rešenja u celoj populaciji **do**
    if $t \leq dmt$ **then**
      Odbacivanje rešenja za koje je uslov $limit \geq exh$ zadovoljen i zamena takvih rešenja slučajnim rešenjima generisanim pomoću jednačine (65).
    end if
  end for
  Spajanje novo kreirane podpopulacijom 1 sa podpopulacijom 2 i generisanje nove populacije veličine N_p
  Evaluiranje nove populacije
  Podešavanje vrednosti S_{max} parametra prema jednačini (56)
  $t++$
end while
return Najbolje rešenje iz populacije

7.3 Testiranje MBO-ABC hibridnog pristupa na standardnim benčmark problemima

Kada se kreira nova metaheuristika, ili se postojeća unapređuje, bilo sitnim modifikacijama, bilo hibridizacijom sa drugim pristupima, uobičajena je praksa da se novi pristup prvo testira na širem opsegu benčmark problema, sa ciljem preciznijeg utvrđivanja performansi. Zbog ovog razloga, hibridizovani MBO-ABC algoritam je prvo testiran na standardnim i poznatim benčmark problemima bez ograničenja. Na ovaj način, prednosti hibridizovane metaheuristike u odnosu na osnovnu implementaciju uzimajući u obzir osnovne indikatore performansi, brzinu konvergencije i kvaliteta rešenja, mogu kvantitativno da se izraze.

U nastavku su prvo prikazane formulacije standardnih benčmark funkcija sa kojima su izvršene simulacije i podešavanja kontrolnih parametara MBO-ABC metaheuristike, a zatim i komparativna analiza sa osnovnim MBO pristupom i drugim vrhunskim metaheuristikama koje su testirane sa istim instancama benčmark problema.

7.3.1 Benčmark funkcije i podešavanje parametara

Hibridini algoritam MBO-ABC testiran je na standardnom skupu od osam benčmark funkcija. Detalji funkcija su dati u Tabeli 1. Radi lakšeg pregleda, svakoj funkciji je dodeljen jedinstveni identifikator (ID).

Da bi se napravila razlika između tipova MBO-ABC parametara, parametri su grupisani u sledeće kategorije: osnovni parametri inicijalizacije, MBO i MBO-ABC opšti kontrolni parametri i MBO-ABC specifični kontrolni parametri.

Tabela 1: *Karakteristike benčmark funkcija korišćenih u simulacijama sa MBO-ABC algoritmom*

ID	Naziv benčmarka	Opseg vrednosti parametra	Optimum	Razdvojivost	Modalitet
F1	Alpine	$[-10,10]$	0.00	separable	multimodal
F2	Brown	$[-1,4]$	0.00	nonseparable	unimodal
F3	Dixon & Price	$[-10,10]$	0.00	nonseparable	multimodal
F4	Fletcher–Powell	$[-\pi,\pi]$	0.00	nonseparable	multimodal
F5	Powell	$[-4,5]$	0.00	separable	unimodal
F6	Rastrigin	$[-5.12,5.12]$	0.00	nonseparable	multimodal
F7	Schwefel 2.21	$[-100,100]$	0.00	nonseparable	unimodal
F8	Zakharov	$[-5,10]$	0.00	nonseparable	unimodal

Grupa osnovnih parametara inicijalizacije uključuje podešavanja koja su korišćena u svim populacionim metaheuristikama, dok ostale dve grupe dalje razdvajaju kontrolne parametre koji su zajednički za oba algoritma (MBO i MBO-ABC), od parametara koji su specifični za MBO-ABC implementaciju.

Jedan od najvećih izazova kada se adaptira ili testira bilo koji algoritam inteligencije rojeva odnosi se na izbor optimalne vrednosti kontrolnih parametara. Skup optimalnih vrednosti kontrolnih parametara bilo koje meteheuristike rojeva zavisi od specifičnog problema, a u većini slučajeva se određuju empirijski izvršavanjem praktičnih simulacija po principu „pokušaja i greške" (eng. trial and error).

U simulacijama su korišćene iste vrednosti osnovnih parametara inicijalizacije i zajedničkih parametara MBO i MBO-ABC algoritama kao i u radu, u kojem je prvi put prikazana osnovna MBO verzija [143]. Autori rada [143] su vršili detaljne empirijske simulacije i odredili su optimalne vrednosti parametra MBO algoritma za probleme globalne optimizacije bez ograničenja.

S druge strane, u slučaju exh i dmt specifičnih kontrolih parametara MBO-ABC algoritma, određene su optimalne vrednosti na osnovu praktičnih simulacija. Obzirom da je MR parametar usvojen iz ABC algoritma, podešena je vrednost ovog parametra na 0.8, kao što je predloženo u radu [52].

Tabela 2: *Podešavanja parametara MBO i MBO-ABC algoritma*

Naziv parametra	**Notacija parametra**	**Vrednost parametra**
Osnovni parametri inicijalizacije		
Ukupan broj rešenja u populaciji	N_p	50
Broj rešenja u podpopulaciji 1	N_{sp1}	21
Broj rešenja u podpopulaciji 2	N_{sp2}	29
Maksimalni broj iteracija	$maxIter$	50
Zajednički kontrolni parametri MBO i MBO-ABC		
Odnos rešenja u podpopulaciji1 prema rešenjima u podpopulaciji2	p	5/12
Period migracije	$peri$	1.2
Najveća dužina koraka	S_{max}	1.0
Stopa podešavanja leptira	BAR	5/12
Specifični kontrolni parametri MBO-ABC algoritma		
Vrednost istrošenosti	exh	4
Okidač za pokretanje mehanizma odbacivanja rešenja	dmt	33
Stopa modifikacije	MR	0.8

Kako je navedeno u Tabeli 2, vrednosti exh i dmt su podešene na 4 i 33, respektivno. Izvodeći praktične simulacije, ustanovljeno je da optimalna ili blizu optimalna vrednost za exh parametar može da se izračuna primenom sledećeg izraza:

$$exh = round(\frac{maxIter}{N_p} \cdot 4). \tag{66}$$

Vrednost *dmt* parametra se računa po formuli $round(maxIter/1.5)$. Konačno, optimalna vrednost za parametar MR je izabran empirijski i ustanovljeno je da se najbolje vrednosti generišu ako je vrednost MR postavljena na 0.8, kao što je i predloženo u radu [52].

7.3.2 Rezultati simulacija i diskusija

Za potrebe komparativne analize i preciznijeg evaluiranja performansi, MBO i MBO-ABC algoritam su testirani na osam benčmark funkcija u 30 nezavisnih puštanja algoritma (eng. run), gde je za svako puštanje, korišćeno različito seme (eng. seed) instance pseudo-slučajnog broja.

Osim toga, svaka funkcija je testirana sa različitim brojem dimenzija, i to u prvom slučaju sa 30 dimenzija ($D = 30$), a u drugom sa 60 ($D = 60$). U svim sprovedenim eksperimentima, zabeležane su najbolja i prosečna vrednost, kao i standardna devijacija, izračunate na osnovu 30 nezavisnih puštanja algoritma.

Komparativna analiza između MBO-ABC hibridne metaheuristike i osnovne verzije MBO algoritma za benčmark probleme sa 30 dimenzija ($D = 30$) je prikazana u Tabeli 3 [121]. Zarad lakše vizuelizacije performansi algoritama, bolji rezultati za sve kategorije testova označeni su bold stilom.

Tabela 3: *Komparativna analiza—Hibridizovani MBO-ABC vs. originalni MBO na benčmark funkcijama sa 30 dimenzija (D = 30)*

ID	Indikator	MBO	MBO-ABC
F1	Najbolji	0.10	**0.08**
	Prosečan	12.93	**7.06**
	St.dev.	17.19	**9.33**
F2	Najbolji	0.02	$\mathbf{7.66 \times 10^{-3}}$
	Prosečan	196.00	**89.05**
	St.dev.	493.80	**153.62**
F3	Najbolji	31.53	**7.32**
	Prosečan	3.57×10^8	$\mathbf{9.52 \times 10^7}$
	St.dev.	$\mathbf{3.54 \times 10^8}$	5.13×10^8
F4	Najbolji	4.63×10^5	$\mathbf{2.23 \times 10^5}$
	Prosečan	8.46×10^5	$\mathbf{5.36 \times 10^5}$
	St.dev.	2.85×10^5	$\mathbf{1.83 \times 10^5}$
F5	Najbolji	**0.67**	0.83
	Prosečan	3.16×10^3	$\mathbf{2.95 \times 10^3}$
	St.dev.	$\mathbf{3.58 \times 10^3}$	3.75×10^3
F6	Najbolji	0.05	**0.03**
	Prosečan	106.00	**87.72**
	St.dev.	84.42	**53.92**
F7	Najbolji	**0.47**	0.51
	Prosečan	45.50	**32.88**
	St.dev.	45.26	**32.15**
F8	Najbolji	31.23	**1.13**
	Prosečan	541.90	**425.39**
	St.dev.	357.30	**303.91**

Na osnovu rezultata iz Tabele 3, očigledno je da MBO-ABC algoritam u proseku ostvaruje bolje rezultate nego originalni MBO pristup. Balans eksploatacije-eksploracije, kao i brzina konvergencije je bolja u slučaju hibridne MBO metaheuristike.

Na primer, u slučaju testa sa funkcijom $F2$, originalan MBO algoritam u ranim iteracijama izvršava eksploataciju na pogrešnom delu prostora pretrage. Međutim, u nekim puštanjima, osnovni MBO uspe da „pogodi" pravo mesto prostora pretrage, ali u kasnijim iteracijama, nije sposoban da izvršava finu pretragu u optimalnom domenu prostora pretrage. U istom testiranju, MBO-ABC algoritam uspeva da pronađe obećavajući region prostora pretrage u ranim iteracijama i izvršavanjem fino

podešene pretrage u ovom regionu u kasnijim iteracijama, uspeva da generiše bolje rešenje od osnovnog MBO algoritma. Sličan zaključak može da se izvuče na osnovu posmatranja srednjih vrednosti za isti benčmark, gde MBO-ABC metaheuristika uspeva da ostvari dva puta bolje srednje vrednosti od osnovne MBO implementacije.

Najznačajnija razlika u performansama MBO-ABC i MBO može da se sagleda ako se uporede rezultati simulacija za $F8$ benčmark. U ovom slučaju, sva tri indikatora, najbolji, srednji i standardna devijacija su značajnije bolji u slučaju MBO-ABC simulacija. Na primer, najbolja vrednost koju generiše MBO-ABC je skoro 30 puta bolja od najbolje ostvarene vrednosti originalnog MBO algoritma.

Kada se sagleda svih osam benčmark funkcija, može da se primeti da je MBO-ABC uspeo da ostvari bolje srednje vrednosti u svakoj izvršenoj simulaciji. Na osnovu ovoga zaključuje se da MBO-ABC pokazuje mnogo bolju brzinu konvergencije nego originalni MBO.

Međutim, za određene indikatore pojedinih instanci testova, MBO ostvaruje bolje rezultate nego MBO-ABC. Takav primer uključuje sledeće: najbolji indikatori za $F5$ i $F7$ testove, kao i indikator standardne devijacije za $F3$ i $F5$ benčmark funkcije. U ovim slučajevima, MBO algoritam je ostvario za neznatno bolje rezultate nego MBO-ABC algoritam.

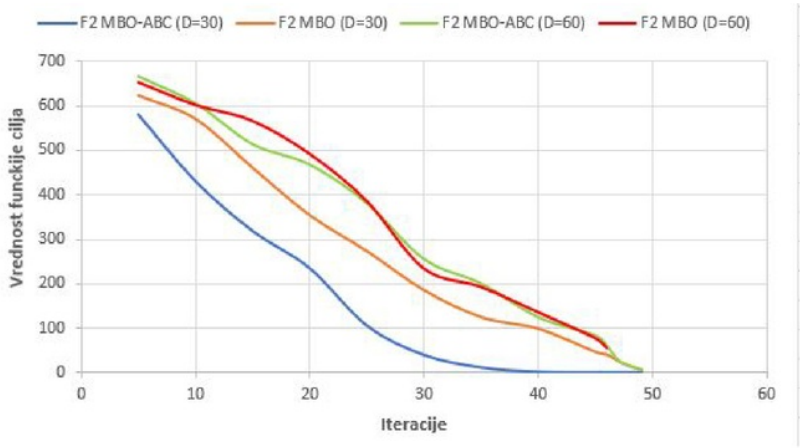
Uprkos činjenici da je osnovni MBO algoritam u nekim slučajevima postigao kvalitetnije rezultate od MBO-ABC algoritma, opšti zaključak na osnovu rezultata izvršenih simulacija, je da MBO-ABC prevazilazi nedostatke nedostatka snage eksploatacije i neadekvatnog balansa između intenzifikacije i diversifikacije originalnog MBO algoritma.

Slično može da se zaključi i za performanse MBO-ABC i MBO na rezultata generisanih iz simulacija u 60-dimenzionalnom prostoru pretrage za iste benčmark funkcije. Rezultati su prezentovani u Tabeli 4 [121]. Slično kao i u testovima u 30-dimenzionom prostoru pretrage, sa ciljem lakše vizuelizacije performansi algoritama, bolji rezultati za sve kategorije testova označeni su bold stilom.

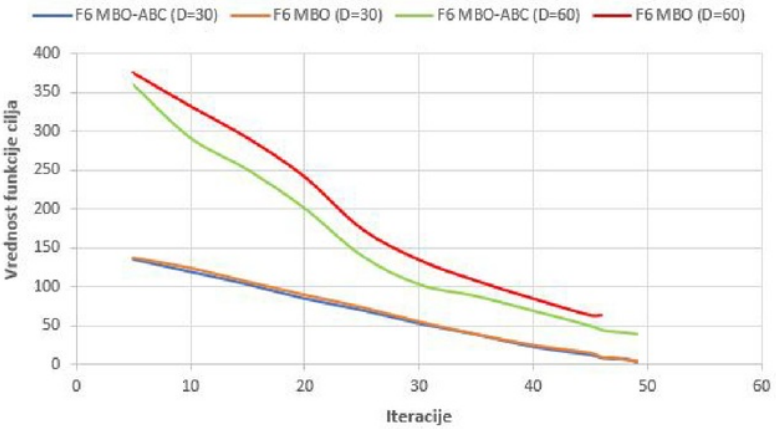
Tabela 4: *Komparativna analiza—hibridni MBO-ABC vs. originalni MBO na benčmark funkcijama sa 60 dimenzija (D = 60)*

ID	Indikator	MBO	MBO-ABC
F1	Najbolji	**0.13**	0.15
	Prosečan	59.21	**41.81**
	St.dev.	61.82	**47.19**
F2	Najbolji	**0.05**	0.19
	Prosečan	5.50×10^{14}	$\mathbf{0.33 \times 10^{9}}$
	St.dev.	3.05×10^{15}	$\mathbf{5.03 \times 10^{9}}$
F3	Najbolji	1.34×10^{4}	**13.29**
	Prosečan	2.50×10^{9}	$\mathbf{1.06 \times 10^{9}}$
	St.dev.	2.12×10^{9}	$\mathbf{1.45 \times 10^{9}}$
F4	Najbolji	$\mathbf{5.20 \times 10^{6}}$	7.43×10^{6}
	Prosečan	7.19×10^{6}	$\mathbf{5.66 \times 10^{6}}$
	St.dev.	$\mathbf{1.33 \times 10^{6}}$	1.95×10^{6}
F5	Najbolji	25.18	**3.03**
	Prosečan	1.29×10^{4}	$\mathbf{1.03 \times 10^{4}}$
	St.dev.	1.22×10^{4}	$\mathbf{1.19 \times 10^{4}}$
F6	Najbolji	60.44	**32.92**
	Prosečan	301.10	**299.44**
	St.dev.	158.90	**140.50**
F7	Najbolji	4.32	**2.23**
	Prosečan	176.20	**145.81**
	St.dev.	**86.66**	99.33
F8	Najbolji	165.70	**13.85**
	Prosečan	5.02×10^{5}	$\mathbf{6.96 \times 10^{3}}$
	St.dev.	1.93×10^{6}	$\mathbf{2.91 \times 10^{4}}$

Grafikoni brzine konvergencije za najbolje puštanje u slučaju testova sa $F2$ i $F6$ benčmark funkcijama sa $D = 30$ i $D = 60$ MBO-ABC i MBO metaheuristika su prikazani na slikama 9 i 10, respektivno.



Slika 9: *Brzina konvergencije MBO-ABC i MBO za F2 benčmark sa $D = 30$ i $D = 60$.*



Slika 10: *Brzina konvergencije MBO-ABC i MBO za F6 benčmark sa $D = 30$ i $D = 60$.*

Na grafikona prikaznog na Slici 9, može da se primeti da je brzina konvergencije u slučaju $F2$ testa bolja kod MBO-ABC pristupa, u odnosu na osnovni MBO algoritam. S druge strane, u slučaju $F6$ testa sa 60 dimenzija (Slika 10), MBO-ABC ostvaruje značajno bolju konvergenciju u odnosu na originalni MBO algoritam. U slučaju 30-dimenzionalnog $F6$ testa, oba pristupa ostvaruju skoro istu brzinu konvergencije.

Sa ciljem daljeg izvođenja empirijskih dokaza za teorijsku pozadinu predloženog MBO-ABC algoritma, najviše ističući balans intenfifikacije-diverzifikacije, sprovedene su simulacije sa promenljivim vrednostima exh i dmt parametara, dok su ostali MBO-ABC parametri podešeni kao u Tabeli 2. Uprkos činjenici da vrednost MR parametra ima uticaj na podešavanja odnosa između intenzifikacije i diverzifikacije, vrednost ovog parametra nije podešavana, s obzirom da je MR usvojen iz ABC

algoritma i podešen kao u [52].

Sa povećanjem *exh* vrednosti parametra, snaga eksploatacije se povećava, dok se u isto vreme intenzitet eksploracije smanjuje, i obrnuto. Kao što je naznačeno ranije, ustanovljeno je da se optimalna vrednost *exh* parametra, sa svim drugim parametrima treba podesiti kao u Tabeli 2, na 4.

Slično kao u slučaju *exh* kontrolnog parametra, kada se vrednost *dmt* parametra povećava, snaga diverzifikacije se smanjuje, dok se istovremeno snaga intenzifikacije procesa pretrage poboljšava. Kao što je naznačeno ranije, takođe je ustanovljeno da se optimalna vrednost *dmt* parametra, sa svim drugim parametrima kao u Tabeli 2, treba podesiti na 33. Sa ovakvim podešavanjima, optimalni balans između diverzifikacije i intenzifikacije MBO-ABC procesa pretrage može da se uspostavi [121].

U Tabeli 5 [121], prikazani su rezultati testiranja MBO-ABC benčmarka za klasu problema globalne optimizacije sa ograničenjima vrednosti parametara, sa promenljivim vrednostima *exh* parametra (za ovaj parametar su redom uzimane vrednosti 3, 4 i 5).

U Tabeli 6 [121], prikazani su takođe rezultati simulacije testiranja MBO-ABC metaheuristike za istu klasu problema sa 30 dimenzija, ali ovog puta sa promenljivim vrednostima parametra *dmt* (za ovaj parametar su redom uzimane vrednosti 32, 33 i 34).

Slično kao i u prethodnim tabelama sa rezultatima, najbolji rezultati iz svih kategorija testova prikazani su zatamljenim stilom.

Na osnovu rezultata prikazanih u navedenim tabelama, može da se vidi da MBO-ABC uspostavlja najbolje performanse u vidu kvaliteta rešenja (najbolje vrednosti) i brzine konvergencije, kada su *exh* i *dmt* vrednosti parametara podešene kao u Tabeli 2. Sa ovakvim podešavanjima, balans između eksploatacije i eksploracije je blizu optimalnog, i MBO-ABC ostvaruje najbolje performanse [121].

Tabela 5: *Eksperimenti sa različitim vrednostima parametra exh MBO-ABC algoritma za benčmark funkcije sa 30 dimenzija*

ID	Indikator	$exh = 3$	$exh = 4$	$exh = 5$
F1	Najbolji	0.11	**0.08**	0.09
	Prosečan	10.54	**7.06**	10.33
	St.dev.	15.22	**9.33**	13.03
F2	Najbolji	9.25×10^{-3}	$\mathbf{7.66 \times 10^{-3}}$	15.02×10^{-3}
	Prosečan	96.15	**89.05**	93.77
	St.dev.	303.14	**153.62**	275.11
F3	Najbolji	16.55	**7.32**	8.29
	Prosečan	1.53×10^{8}	$\mathbf{9.52 \times 10^{7}}$	1.01×10^{8}
	St.dev.	8.99×10^{8}	$\mathbf{5.13 \times 10^{8}}$	5.28×10^{8}
F4	Najbolji	3.29×10^{5}	$\mathbf{2.23 \times 10^{5}}$	5.06×10^{5}
	Prosečan	7.66×10^{5}	$\mathbf{5.36 \times 10^{5}}$	7.52×10^{5}
	St.dev.	2.32×10^{5}	$\mathbf{1.83 \times 10^{5}}$	2.68×10^{5}
F5	Najbolji	0.96	**0.83**	0.87
	Prosečan	5.05×10^{3}	$\mathbf{2.95 \times 10^{3}}$	3.23×10^{3}
	St.dev.	4.02×10^{3}	$\mathbf{3.75 \times 10^{3}}$	3.81×10^{3}
F6	Najbolji	0.04	**0.03**	0.04
	Prosečan	96.91	**87.72**	93.50
	St.dev.	67.47	**53.92**	58.00
F7	Najbolji	0.53	**0.51**	0.52
	Prosečan	35.51	**32.88**	35.67
	St.dev.	36.27	**32.15**	34.13
F8	Najbolji	5.92	**1.13**	7.35
	Prosečan	471.20	**425.39**	485.50
	St.dev.	337.19	**303.91**	346.58

Tabela 6: *Eksperimenti sa različitim vrednostima parametra dmt MBO-ABC algoritma za benčmark funkcije sa 30 dimenzija*

ID	Indikator	$dmt = 32$	$dmt = 33$	$dmt = 34$
F1	Najbolji	0.08	0.08	0.09
	Prosečan	9.24	**7.06**	10.02
	St.dev.	13.77	**9.33**	13.85
F2	Najbolji	10.13×10^{-3}	$\mathbf{7.66 \times 10^{-3}}$	9.12×10^{-3}
	Prosečan	99.37	**89.05**	98.62
	St.dev.	325.19	**153.62**	307.66
F3	Najbolji	14.09	**7.32**	8.05
	Prosečan	2.57×10^{8}	$\mathbf{9.52 \times 10^{7}}$	9.85×10^{7}
	St.dev.	6.33×10^{8}	$\mathbf{5.13 \times 10^{8}}$	6.71×10^{8}
F4	Najbolji	3.51×10^{5}	$\mathbf{2.23 \times 10^{5}}$	5.22×10^{5}
	Prosečan	6.39×10^{5}	$\mathbf{5.36 \times 10^{5}}$	6.13×10^{5}
	St.dev.	1.92×10^{5}	$\mathbf{1.83 \times 10^{5}}$	2.09×10^{5}
F5	Najbolji	0.91	**0.83**	0.85
	Prosečan	5.11×10^{3}	$\mathbf{2.95 \times 10^{3}}$	3.07×10^{3}
	St.dev.	4.81×10^{3}	$\mathbf{3.75 \times 10^{3}}$	3.78×10^{3}
F6	Najbolji	0.06	**0.03**	0.04
	Prosečan	99.52	**87.72**	90.30
	St.dev.	69.55	**53.92**	56.82
F7	Najbolji	0.53	**0.51**	0.53
	Prosečan	34.17	**32.88**	35.22
	St.dev.	36.99	**32.15**	33.82
F8	Najbolji	4.12	**1.13**	6.92
	Prosečan	461.25	**425.39**	483.19
	St.dev.	350.92	**303.91**	352.34

7.4 Testiranje MBO-ABC metaheuristike na problemu raspoređivanja poslova na klaudu

U ovom poglavlju prikazani su rezutati pratičnih simulacija MBO-ABC metaheuristike za rešavanje praktičnog NP teškog problema raspoređivanja poslova i balansiranja opterećenja na klaudu.

Na početku ovog poglavlja opisano je prilagođavanje (adaptiranje) hibridnog MBO-ABC pristupa za ovaj problem i CloudSim simulator, koji je korišćen u simulacijama. Zatim su prikazani rezultati testiranja na modelu raspoređivanja poslova na klaudu sa jednom funkcijom cilja korišćenjem veštačkih i realnih skupova podataka, kao i

komparativna analiza sa osnovnim MBO i drugim heuristikama i metaheuristikama korišćenjem veštačkog i realnog skupa podataka.

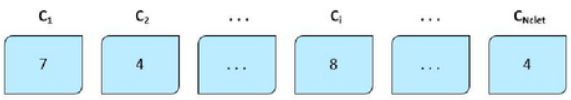
Formulacija modela raspoređivanja poslova na klaudu sa jednom funkcijom cilja data je u Poglavlju 4.2.

7.4.1 Prilagođavanje hibridne MBO metaheuristike za model raspoređivanja poslova

Jedan od najvećih izazova u procesu adaptacije bilo koje metaheuristike za rešavanje specifičnog problema jeste kako da se kodiraju kandidat rešenja problema. Na osnovu formulacije problema raspoređivanja poslova na klaudu sa jednom funkcijom cilja, date u Sekciji 4.2, ako postoji N_{clet} poslova i N_{vm} virtuelnih mašina, veličina prostora pretrage i broj mogućih raspodela zahteva na virtuelne mašine su dati izrazom $(N_{vm})^{N_{clet}}$.

S obzirom da osnovni MBO algoritam nije implementiran za ovaj problem, za potrebe simulacija i komparativne analize, implementirani su i osnovni i hibridni MBO. U implementaciji obe verzije MBO algoritma, osnovne i hibridizovane, potencijalno rešenje se reprezentuje kao skup poslova koji treba da se izvrše, gde svakom zahtevu (poslu) treba da se dodeli odgovarajuća virtuelna mašina koja može efikasno da ga obradi.

Svaka individua u populaciji se kodira kao niz veličine N_{clet}, gde svaki element niza dobija vrednost iz opsega [1,N_{vm}]. Primer kandidat rešenja sa N_{clet} broja poslova i $N_{vm} = 8$ broj virtuelnih mašina je dat na Slici 11 [121]. U ovom primeru, zahtev C_1 se dodeljuje za izvršavanje na virtuelnoj mašini sa $VM_{ID} = 7$, zahtev C_i će biti izvršen na virtuelnoj mašini sa $VM_{ID} = 8$, dok će poslovi C_2 i $C_{N_{clet}}$ biti obrađeni na virtuelnoj mašini sa $VM_{ID} = 4$.



Slika 11: *Primer šeme kodiranja rešenja*

Ako se kao drugi primer uzme u obzir 10 poslova i 4 virtuelne mašine, potencijalno rešenje može da se presentuje niz [3 2 3 1 4 2 4 2 2 1].

U fazi inicijalizacije svakog puštanja algoritma, N_p broj individua su nasumično generisane, gde se svaka individua x_i prezentuje kao niz veličine N_{clet} ($x_i = x_{i,1}$, $x_{i,2}$, $x_{i,3}$, ..., $x_{i,N_{clet}}$). Rešenja (individue) se generišu korišćenjem sledećeg izraza:

$$x_{i,j} = round(\phi \cdot (ub - lb) + lb), \tag{67}$$

gde su za sve komponente rešenja $j = 1, 2, 3, ..., N_{clet}$, gornja granica ($ub$) i donja granica ($lb$) podešene na N_{vm} i 1, respektivno. Notacija ϕ predstavlja pseudo-slučajan broj iz intervala $[0, 1]$.

Obzirom da je prostor pretrage relativno veliki, nije postojala potreba da se MBO-ABC metaheuristika prilagodi za probleme celobrojnog programiranja. U jednačini (67), kao i u jednačini (54) i (55), rezultati su zaokruženi na najbliži celi broj korišćenjem jednostavne *round* funkcije.

7.4.2 Prikaz CloudSim platforme

Simulacije raspoređivanja poslova na klaudu izvršene su u CloudSim toolkit okruženju. Izvršena su dva tipa testova (simulacija). U prvom testu je korišćen veštački skup podataka koji je nasumično generisan u okviru CloudSim platforme. U drugom eksperimentu, korišćeni su pravi (realni) skupovi podataka koji su preuzeti iz logova jedne od produkcionih klaud infrastruktura.

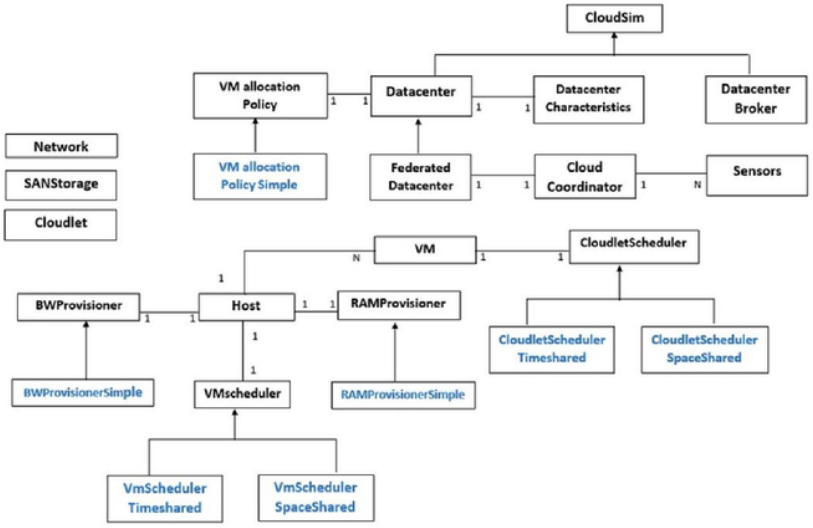
ClaudSim je samostalni okvir (eng. framework) i kao takav predstavlja proširiv simulacioni set alata za simuliranje klaud okruženja i omogućava modeliranje i simulaciju klaud računarskih sistema i klaud aplikacija [21]. CloudSim toolkit softver, koji pripadi grupi softvera otvorenog koda (eng. open source), u sklopu licence Apache verzije 2.0, razvijen je u Java tehnologiji u laboratoriji za klaud računarstvo i distribuirane sisteme, i široko se primenjuje za istraživanje, kako od strane akademske zajednice, tako i od strane industrijskih entiteta (korporacija). CloudSim može besplatno da se preuzme sa sledećeg URL-a: `https://github.com/Cloudslab/cloudsim/releases`.

CloudSim obezbeđuje kontrolisano eksperimentalno okruženje za kreiranje i modeliranje klaud entiteta, može da se proširi u kontekstu uključivanja korisničko-definisanih polisa za raspoređivanje i alokaciju, pruža mrežnu arhitekturu i infrastrukturu koja se lako modifikuje, a takođe omogućava primenu tehnologiju virtuelizacije,

obezbeđuje softversku emulaciju host računara, mreže i mehanizam rezervisanja aplikacija [21]. Sveobuhvatna biblioteka CloudSim-a sastoji se iz 12 paketa, koji mogu da se koriste za simuliranje različitih tipova scenarija klaud računarstva. Najbitniji paket je CloudSim koji sadrži klase za modeliranje različitih klaud entiteta kao što su Datacenter, Cloudlet, Host, VM i DatacenterBroker.

Pregled CloudSim dizajn dijagrama klase i arhitekture je dat na Slici 12 (slika preuzeta i adaptirana iz [21]).

U nastavku su ukratko opisane osnovne komponente CloudSim platforme.



Slika 12: *CloudSim arhitektura*

Cloudlet predstavlja klasu koja modelira servise klaud orijentisanih aplikacija, kao što su isporuka sadržaja i tok poslova, ali takođe i drugih poslova koji se preuzimaju od krajnjih korisnika za obradu u klaud sistemu. Pre početka svake simulacije, osnovni parametri kao što su dužina instrukcija i metod transfera podataka trebaju da se definišu, što se radi u okviru ove klase. Po potrebi, klasa cloudlet može da se proširi na različite tipove modela i aplikativnih servisa.

CloudletScheduler je abstraktna klasa koja se koristi za implementiranje različitih polisa koje definišu način na koji se snaga obrade virtuelnih mašina deli između poslova. U osnovnoj CloudSim biblioteci, dva tipa ovih polisa su dostupne, i to: space-shared (*CloudletSchedulerSpaceShared*) and time-shared (*CloudletScheduler-TimeShared*). Istraživači mogu da prošire ovu klasu za modeliranje drugih polisa,

prema postrebama istraživanja i korišćenog modela.

Klasa *Datacenter* modelira hardversku arhitekturu, na kojoj se svaka virtuelna mašina izvršava. U ovoj klasi je enkapsulirana polisa za definisanje karakteristika platforme host računara, koja može da bude homogena i ili heterogena. Homogeni host deli istu konfiguraciju hardvera (broj i tip CPU i CPU jezgara, memoriju, kapacitet skladištenja). U heterogenom host okruženju, različiti hostovi imaju različite karakteristike fizičkog hardvera. Takođe je korisno da se napomene da svaka *Datacenter* klasa instancira komponentu interfejsa za rezervisanje aplikacija, koja implementira grupu polisa za mrežni protok, memoriju i alokaciju skladišta host računara i virtuelnih mašina.

Klasa *DatacenterBroker* modelira entitet brokera koji posreduje između SaaS i CSP. Broker radi u sklopu SaaS provajdera. Kada se prihvati zahtev koji dolazi sa strane krajnjih korisnika, broker otvkriva odgovarajući CSP servis, tako što šalje upit servisu klaud informacija (eng. cloud information service-CIS), gde se zatim izvršava proces online pregovaranja u vezi alokacije resursa i/ili servisa koji moraju da zadovoljavaju kvalitet usluga aplikacije (eng. Quality of Service - QoS) [21]. Ova klasa može da se proširi sa ciljevima daljeg razvoja i testiranja polisa brokera.

Klasa *DatacenterCharacteristics* enkapsulira informaciju koja se odnosi na konfiguraciju različitih resursa data centara. *Host* je glavna CloudSim komponenta koja modeluje fizički resurs. Ova klasa sadrži različite bitne podatke o karakteristikama fizičkog resursa, kao što je količina dostupne RAM memorije i kapacitet skladištenja, listu i tip elemenata obrade (jezgra), informaciju o polisi koja se primenjuje za daljenje snage obrade podataka između virtuelnih mašina, polise deljenja protoka mreže i memorije za virtuelne mašine. Klasa *VmScheduler* je abstraktna klasa koju *Host* komponenta implementuje, koja se koristi za modelovanje polise deljenog prostora (eng. space-shared) i deljenog vremena (eng. time-shared) za alokaciju jezgara procesora fizičkog hosta na virtuelne mašine.

Svaka *Host* komponenta CloudSim-a upravlja i hostuje jednu ili više virtuelnih mašina. U klasa *Vm* moguće je podesiti sledeće važne atribute svake virtuelne mašine: brzinu obrade, pristupna memorija, veličina prostora skladištenja podataka, kao i internu polisu rezervisanja koja se nasleđuje apstraktnu komponentu *CloudletScheduler* [21].

Klasa *VmmAllocationPolicy* predstavlja apstraktnu klasu koja modelira polisu rezervisanja koja se koristi za alociranje virtuelnih mašina na host računare. Najvažnija uloga ove klase je izbor raspoloživog host računara iz data centra, koji zadovoljava minimalne zahteve za obezbeđivanje virtuelne mašine. Ovi zahtevi se pre svega odnose na potrebnu memoriju, snagu obrade i skladištenje.

U osnovnom CloudSim instalacionom paketu, uključeni su neki primeri generisanja i izvršavanja CloudSim simulacija. Instance svih primera su podeljene u nekoliko kategorija: osnovni primeri, sumulacije mrežnog okruženja, simulacije sa heterogenim virtuelnim mašinama i primeri sa promenljivim skladišnim prostorom. Ovi primeri pružaju sasvim dovoljno informacija za istraživače koji žele da izvrše simulacije u CloudSim ekperimentalnom okruženju.

Proces generisanja CloudSim simulacija (životni ciklus simulacija) može da se sumira u osam koraka: inicijalizacija CloudSim paketa, kreiranje data centara, kreiranje brokera, generisanje poslova, kreiranje virtuelnih mašina, pokretanje simulacije (automatski proces koji se izvršava putem simulacija diskretnih događaja), zaustavljanje simulacije i štampanje rezultata (vizuelizacija izlaznih rezultata) [21].

U eksperimentalnim simulacijama za potrebe ove disertacije, korišćena je verzija 3.0.3 CloudSim platforme. Da bi se u okviru simulatora implementirale metaheuristike inteligencije rojeva, koje imaju ulogu komponentu raspoređivača poslova, potrebno je bilo modifikovati neke od osnovnih CloudSim klasa.

U svim simulacijama, korišćena je sledeća računarska platforma: Intel Core™ i7-8700K CPU sa 32 GB RAM memorije, Windows 10 operativni sistem i Java SE Development Kit 12.0.1.

7.4.3 Simulacije sa veštačkim skupom podataka

Kao što je već i navedeno, MBO-ABC metaheuristika je testirana sa veštačkim i realnim skupovima podataka. Za potrebe prvog skupa eksperimenata, korišćeni su veštački skupovi podataka generisani u CloudSim okruženju i model raspoređivanja poslova na klaudu sa jednom funkcijom cilja. Detaljan prikaz ovog modela dat je u Poglavlju 4.2.

U radu objavljenom u jednom vrhunskom međunarodnom časopisu [165], korišćen je isti model sa istim skupovima podataka, gde se prikazani rezultati testiranja više

vrhunskih heurističkih i metaheurističkih metoda. Da bi se performanse hibridnog MBO-ABC algoritama uporedile sa kvalitetom generisanih rešenja drugih vrhunskih pristupa, u simulacijama sa veštačkim skupom podataka, MBO-ABC je testiran u istim eksperimentalnim uslovima, kao i metode koje su prikazane u [165].

Izvršene su simulacije sa različitim brojem poslova, u rasponu od 100 do 600 kao u [165]. Rezultati svake instance testa su evaluirani u 100 nezavisnih puštanja algoritma. Na ovaj način je uspostavljeno smanjenje uticaja nasumičnosti prikupljenih rezultata, kako bi se izvršavala preciznija evaluacija performansi predloženih metaheuristika. Za funkciju cilja korišćena je vrednost makespan indikatora.

U fazi inicijalizacije svakog puštanja algoritma, generisano je klaud okruženje sa slučajnim karakteristikama, koje su prikazane u Tabeli 7 [121]. Kao što se može videti iz Tabele 7, za svako nezavisno puštanje algoritma, kreirano je nasumično klaud okruženje, gde su vrednosti za CPU MIPS svake virtuelne mašine uzete iz opsega [1860,2660], dužina zahteva poslova varira u opsegu [400–1000], itd. Ostali atributi klaud okruženja, poput troškova računarske mreže, troškovi transfera sa jednog na drugi entitet, itd., uzete su podrazumevane (eng. default) vrednosti iz CloudSim simulatora, koje su definisane u DatacenterCharacteristics CloudSim entitetu.

Kada se govori o konfiguraciji CloudSim okruženja, potrebno je posebno da se istaknu polise. U ovom okruženju postoje dva osnovna tima polisa. Prva polisa, polisa alokacije virtuelnih mašina (eng. vm allocation policy) se odnosi na alokaciju (dodeljivanje) virtuelnih mašina host (fizičkim) računarima. Druga polisa, polisa alokacije zadataka (eng. task allocation policy) se odnosi na dodeljivanje (mapiranje) zadataka na virtuelne mašine. Oba tipa polisi mogu da budu bazirane na vremenu (eng. time-shared) ili na prostoru (eng. space-shared).

U izvršenim simulacijama, stopa pristizanja poslova na izvršavanje (eng. cloudlet arrival rate) podešena je na 10 zadataka u sekundi, kao i u [165]. Da bi se promenila stopa pristizanja poslova na obradu u klaud okruženje, modifikovana je *DatacenterBroker* CloudSim klasa entiteta, tačnije, metoda *submitCloudlet()* u okviru ove klase.

U radu [165], unapređena verzija ACO metaheuristike, nazvana ACO orijentisana na performanse budžeta (eng. budget performance ACO - PBACO), za problem raspoređivanja poslova u klaud okruženju, je prezentovana i testirana pod istim

eksperimentalnim uslovima kao i MBO-ABC algoritam.

Da bi se prikazala što realnija komparativna analiza, kako sa PBACO algoritmom, tako i sa drugim heuristikama i metaheuristikama, prikazanih u [165], vrednosti vrednosti parametara N_p i $maxIter$ su podešene na 20 i 50, respektivno, čime se ostvaruje ukupan broj od 1000 evaluacija funkcije cilja ($20 \cdot 50 = 1000$). Algoritam inteligencije rojeva PBACO je testiran sa 10 mrava u populaciji i 100 iteracija, što isto kao i u slučaju MBO-ABC algoritma, generiše ukupan broj od 1000 evaluacija objektne funkcije.

Vrednost exh parametra MBO-ABC hibrida, koja je izračunata na osnovu jednačine (66), podešena je na 10, dok je vrednost dmt parametra podešena na 33, korišćenjem istih jednačina kao i u simulacijama globalne optimizacije bez ograničenja prikazanih u Poglavlju 7.3.1.

Ostale vrednosti parametara MBO i MBO-ABC metaheuristika postavljene su na vrednosti koje su prikazane u Tabeli 2.

Tabela 7: *Parametri slučajno generisanog CloudSim okruženja u simulacijama sa veštačkim skupom podataka (MBO-ABC algoritam)*

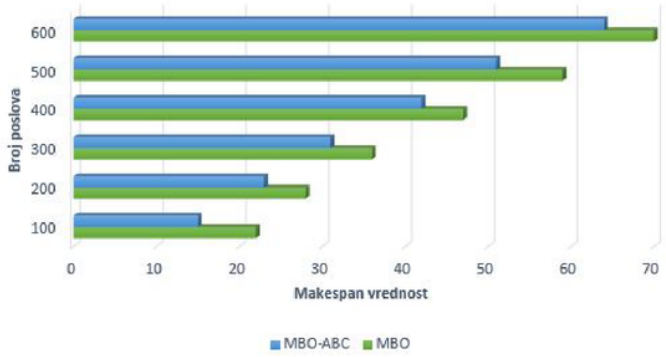
CloudSim Entitet	**Parametar**	**Vrednost**
Poslovi (zahtevi)	broj poslova	100–600
	dužina poslova	400–1000 (jedinica: MI)
	veličina fajla	200-1000 (jedinica: MB)
	veličina izlaza	20–40 (jedinica: MB)
Virtuelne mašine	broj VM-a	10
	CPU mogućnosti	1860–2660 (jedinica: MIPS)
	RAM	4096 (jedinica: MB)
	mrežni protok	100 (jedinica: Mbps)
	kapacitet skladištenja	10 (jedinica: GB)
	polisa raspoređivanja poslova	bazirana na vremenu
	VMM	Xen
	O.S.	Linux
	broj CPU-a	1
Host	broj host-ova	2
	RAM	32 (jedinica: GB)
	kapacitet skladištenja	1 (jedinica: TB)
	mrežni protok	10 (jedinica: Gbps)
	polisa raspoređivanja VM	bazirana na vremenu
Datacentar	broj datacentara	2

Da bi se kvantifikovala unapređenja MBO-ABC metaheuristike u odnosu na osnovni MBO, prvo je urađena komparacija između MBO-ABC i MBO algoritma.

Oba algoritma su testirana na istim instancama problema i pod istim eksperimentalnim uslovima. Kao što je već i rečeno, obzirom da prethodne implementacije osnovnog MBO algoritma za ovu klasu problema nisu raspoložive, za potrebe ovog istraživanja implementiran je i adaptiran i osnovni MBO pristup. Rezultati simulacija i komparativna analiza su prikazani na Slici 13 [121].

Na osnovu rezultata prikazanih na Slici 13, zaključuje se da MBO-ABC postiže značajno bolje performanse od originalnog MBO u svim instancama simulacija. Na primer, u slučaju testa sa 100 poslova, MBO-ABC ostvaruje smanjanje makespan indikatora od 31.81% u odnosu na originalni MBO algoritam. Poboljšanja performansi u testovima sa 600 poslova je 8.57%, dok je ista metrika u simulacijama sa 300 i 400 zadataka bolja za oko 13%.

Procentualno istražena unapređenja MBO-ABC algoritma u odnosu na osnovni MBO za sve instance testova sa veštačkim skupom podataka su sumirane u Tabeli 8 [121].

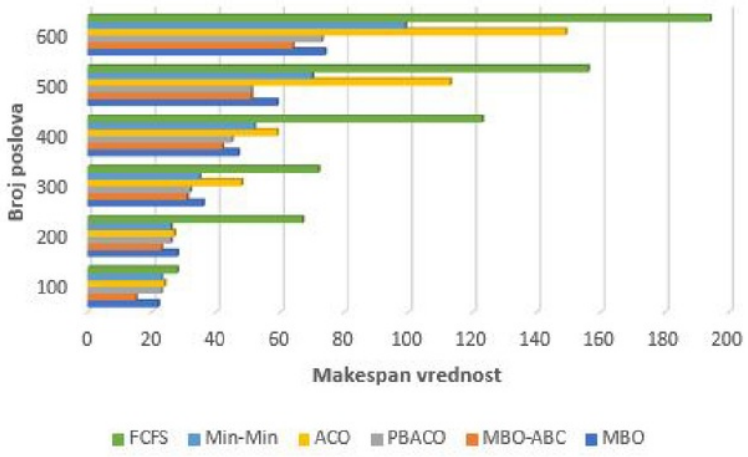


Slika 13: *Komparativna analiza makespan indikatora — MBO-ABC vs. MBO za stopu pristizanja od 10 poslova u sekundi sa veštačkim skupom podataka*

Tabela 8: *Unapređenje makespan vrednosti u odnosu na MBO u simulacijama sa veštačkim skupom podataka*

Broj poslova	Poboljšanja makespan vrednosti
100	+31.81%
200	+17.85%
300	+13.88%
400	+10.63%
500	+13.56%
600	+8.57%

Komparativna analiza između MBO-ABC i Min-Min, first come first serve (FCSF) heuristike i ACO i PBACO metaheuristike su prikazane na Slici 14 [121]. Rezultati za Min-Min, FCFS, ACO i PBACO su preuzeti iz [165]. Originalni MBO je takođe uključen u komparativnu analizu.



Slika 14: *Komparativna analiza makespan indikatora —MBO-ABC vs. ostali pristupiza stopu pristizanja od 10 poslova u sekundi sa veštačkim skupom podataka*

Na osnovu komparativne analize prikazane na Slici 14, može da se zaključi da su od svih pristupa najbolji kvalitet rešenja generiše MBO-ABC metaheuristika, druga po redu je PBACO, dok je na trećem mestu originalni MBO. Samo u slučaju testa sa 500 poslova, PBACO ostvaruje istu vrednost makespan funkcije cilja kao i MBO-ABC. Originalna MBO metaheuristika u proseku ostvaruje za nijansu bolje performanse nego osnovni ACO i Min-Min heuristika. Heuristika FCFS ostvaruje najgore performanse u odnosu na sve testirane pristupe.

Povećanje performansi (smanjivanje vrednosti makespan funkcije cilja) izražene u procentima, MBO-ABC u odnosu na druge heuristike i metaheuristike, koje su uključene u komparativnu analizu, prikazano je u Tabeli 9 [121].

Tabela 9: *Unapređenje makespan vrednosti u odnosu na druge metaheuristike u simulacijama sa veštačkim skupom podataka*

Broj poslova	PBACO	ACO	Min-Min	FCFS
100	+34.78%	+37.50%	+34.78%	+17.85%
200	+11.53%	+14.81%	+11.53%	+61.19%
300	+3.12%	+35.41%	+11.42%	+55.55%
400	+6.66%	+28.81%	+19.23%	+63.41%
500	+0.00%	+54.86%	+27.14%	+67.30%
600	+12.32%	+57.04%	+35.35%	+62.37%

7.4.4 Simulacije sa realnim skupom podataka

Sa ciljem daljeg testiranja robusnosti, kvaliteta rešenja i skalabilnosti predložene MBO-ABC metaheuristike, simulirano je homogeno klaud okruženje, gde sve virtuelne mašine imaju iste karakteristike, sa različitim brojem poslova koje dospevaju na obradu u klaud okruženje (od 100 do 2000). Za potrebe ovih simulacija, korišćene su vrednosti (atributi) poslova koje su generisane iz realnog klaud okruženja.

Karakteristike zahteva (poslova) koje su korišćene u izvršenim simulacijama preuzeti su iz Švajcarske standardne formatirane evidencije logova visokih performansi računarstva—HPC2N (High Performance Computing Center North). Ovaj log sadrži informacije za 527,371 poslova i 240 CPU resursa prikupljenih u periodu od 2002. do 2006. godine. Preuzet je tzv. *workload trace log* fajl *HPC2N-2002-2.2-cln.swf* sa sledećeg URL-a: `http://www.cs.huji.ac.il/labs/parallel/workload/l_hpc2n/`. Svaki red u preuzetom fajlu označava skup karakteristika jednog posla.

Iz prve kolone *workload trace log* fajla preuzet je ID zadatka, dok su vrednosti za dužinu zadatka (izražene u jedinicama milion intrukcija), i broj potrebnih procesirajućih elemenata za izvršavanja zadtaka (eng. processing element - PE) izvučeni iz četvrte i osme kolone, respektivno.

U izvršenim simulacijama, korišćeno je deset virtuelnih mašina sa istim konfiguracijama, jedan data centar sa osnovnim CloudSim karakteristikama i dva fizička hosta. Isto okruženje, kao skup podataka korišćen je u [34]. U [34], hibridizovana MS metaheuristika sa diferencijalnom evolucijom (eng. moth search algorithm hybridized with differential evolution - MSDE) je prilagođena i testirana za problem raspoređivanja poslova.

Sa ciljem realnije komparativne analize sa MSDE pristupom, veličina populacije

MBO-ABC i MBO metaheuristike N_p su podešene na 30 sa maksimumom od 1000 iteracija u jednom puštanju algoritma. Isti kontrolni parametri su korišćeni u [34].

Kao u simulacijama sa veštačkim skupom podataka, vrednost *exh* parametra je određena na osnovu jednačine (66) i podešena na 133, dok je vrednost *dmt* parametra postavljena na 667, korišćenjem istih izraza, prikazanih u Sekciji 7.3.1). Ostali MBO-ABC i MBO parametri postavljeni su kao što je navedeno u Tabeli 2.

Karakteristike virtuelnih mašina i hostova korišćeni u simulacijama su sumirani u Tabeli 10 [121].

Tabela 10: *CloudSim parametri za simulacije sa realnim skupom podataka (MBO-ABC algoritam)*

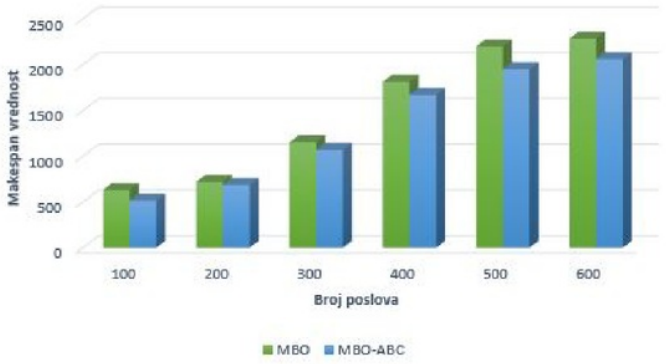
CloudSim Entitet	**Parametar**	**Vrednost**
VM	broj VM	10
	CPU mogućnosti	9726 (jedinica: MIPS)
	RAM	512 (jedinice: MB)
	mrežni protok	1000 (jedinica: Mbps)
	kapacitet skladištenja	10 (jedinica: GB)
	polisa raspoređivanja poslova	bazirana na vremenu
	VMM	Xen
	O.S.	Linux
	broj CPU	1
	CPU tip	Pentium 4 Extreme Edition
Host1	RAM	3 (jedinice: GB)
	CPU tip	Intel Core 2 Extreme X6800
	Broj jezgara (PEs)	2
	CPU mogućnosti	27079 (jedinica: MIPS)
	kapacitet skladištenja	1 (jedinica: TB)
	mrežni protok	10 (jedinica: Gbps)
	polisa raspoređivanja VM	bazirana na prostoru
Host2	RAM	3 (jedinica: GB)
	CPU tip	Intel Core i7 Extreme Edition 3960X
	Broj jezgara (PEs)	6
	CPU mogućnosti	177730 (jedinica: MIPS)
	kapacitet skladištenja	1 (jedinica: TB)
	mrežni protok	10 (jedinica: Gbps)
	polisa raspoređivanja VM	bazirana na prostoru

Kao u prethodnom testu, za svaku instancu testa (broj poslova), metaheuristike su izvršene u 100 nezavisnih puštanja algoritma i izračunata je prosečna vrednost najbolje makespan funkcije cilja i stepen indikatora neraspoređenosti opterećenja između virtuelnih mašina (eng. degree of imbalance indicator - DI). Više o DI indikatoru pogledati u Poglavlju 4.2.

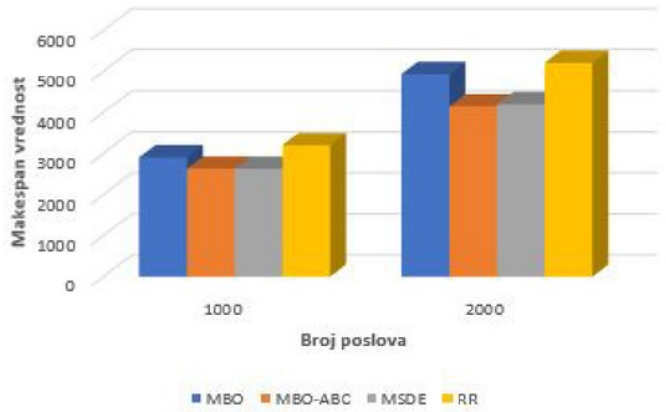
Za razliku od prethodnog testa, nije modifikovana stopa pristizanja poslova u

DatacenterBroker klasi CloudSim okruženja. Obzirom da su u *workload trace log HPC2N-2002-2.2-cln.swf* fajlu dostupne karakteristike 527,371 zahteva, u svakom puštanju algoritma, zahtevi su birani nasumično.

Isto kao i u slučaju prethodnog testa, sa 100 puštanja algoritama za svaku instanci testa, minimizovan je faktor slučajnosti u evaluiranju performansi algoritama. Komparativna analiza između MBO-ABC i originalnog MBO na manjem broju poslova (100–600) i većem broju poslova (1000 i 2000) su prikazani na slikama 15 [121] i 16 [121], respektivno.



Slika 15: *Komparativna analiza—MBO-ABC vs. MBO sa HPC2N bazom realnih podataka sa manjem brojem poslova*



Slika 16: *Komparativna analiza—MBO-ABC vs. MBO sa HPC2N bazom realnih podataka sa manjem brojem poslova*

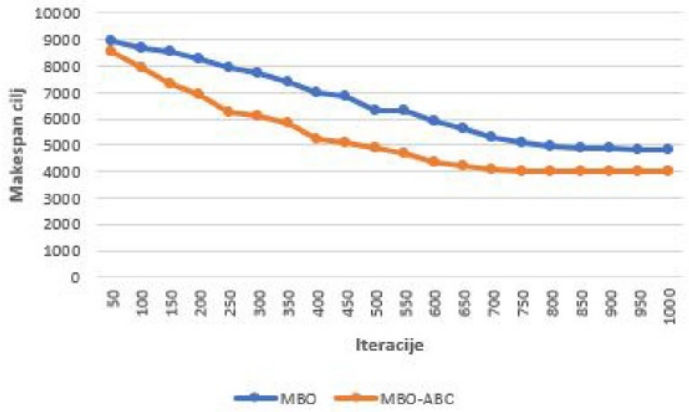
Na osnovu prikazanih rezultata, zaključuje se da MBO-ABC ostvaruje bolje performanse tako što generiše bolje vrednosti makespan funkcije cilja, za oba test

seta, i sa malim i sa većim brojem poslova. Poboljšanja MBO-ABC u odnosu na originalan MBO su prikazana i u procentnim vrednostima u Tabeli 11 [121].

Tabela 11: *Poboljšanja MBO-ABC u odnosu na originalan MBO u simulacijama sa HPC2N realnim skupom podataka*

Broj poslova	Makespan poboljšanja
100	+18.25%
200	+4.72%
300	+7.19%
400	+7.83%
500	+11.27%
600	+9.75%
1000	+9.61%
2000	+16.03%

Grafikon brzine konvergencije MBO-ABC i MBO za instancu problema sa 2000 poslvoa u nekim od boljih puštanja je dat na Slici 17 [121].



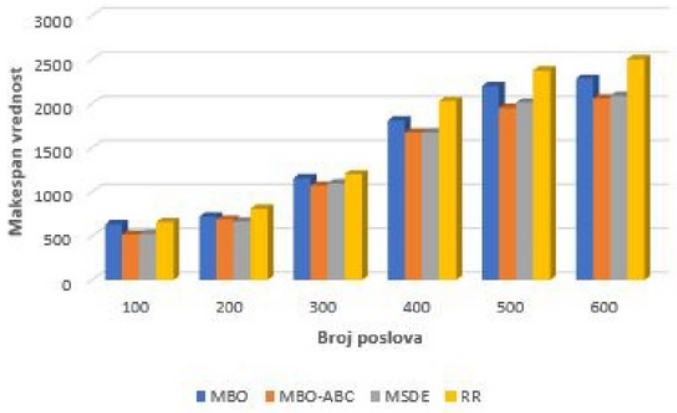
Slika 17: *MBO-ABC vs. MBO brzina konvergencije za instancu problema sa 2000 poslova sa HPC2N realnim skupom podataka*

Na osnovu prikazanog grafa brzine konvergencije (Slika 17), ono što je potrebno izdvojiti jeste to što u ranim iteracijama, MBO-ABC brže konvergira nego originalna MBO metaheuristika. Ovo znači da je mehanizam zamene istrošenih rešenja, koji je adaptiran iz ABC algoritma, poboljšao balans između intenzifikacije i diverzifikacije originalnog MBO, i u ranim iteracijama izvršavanja algoritma, MBO-ABC se ne zaglavljuje u nekom od suboptimalnog domena prostora pretrage. Takođe, izvodeći preciznu analizu, može da se primeti da je MBO-ABC već u iteracijama 650 i 700

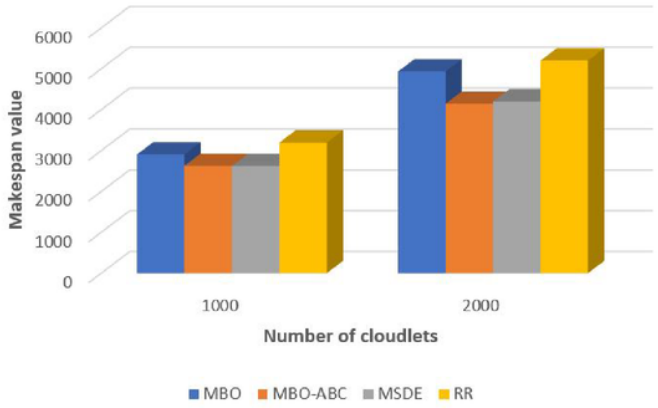
počeo da izvršava finu pretragu u obećavajućem regionu prostora pretrage.

Komparativna analiza rezultata za manji i veći brojeva poslova sa MSDE metaheuristikom i RR heuristikom, prikazana je na slikama 18 [121] i 19 [121], respektivno. U analizu je uključen i originalan MBO.

Analizirajući prikazane rezultate, vidi se da su performanse originalnog MBO pristupa značajno lošije od performansi MSDE metaheuristike, ali bolje od kvaliteta rešenja koja generiše heuristika RR. S druge strane, hibridizovan MBO-ABC u proseku ostvaruje bolje rezultate (veći kvalitet rešenja) nego MSDE metaheuristika. U simulacijama sa 100, 300, 400, 500, 600 i 2000 poslova, MBO-ABC se pokazao da je bolji od MSDE, dok je MSDE nadmašio MBO-ABC u eksperimentima sa 200 i 1000 poslova. Razlog za uključivanje heuristike RR u komparativnu analizu je dokazivanje superiornosti metaheuristika u odnosu na heuristike kada se rešavaju ovakvi tipovi problema.



Slika 18: *Komparativna analiza—MBO-ABC vs. MSDE vs. RR u simulacijama sa HPC2N realnim skupom podataka za manji broj poslova*



Slika 19: *Komparativna analiza—MBO-ABC vs. MSDE vs. RR u simulacijama sa HPC2N realnim skupom podataka za veći broj poslova*

Unapređenja koja generiše MBO-ABC u odnosu MSDE i RR, izražena u procentima, data su u Tabeli 12 [121].

Tabela 12: *Komparativna analiza MBO-ABC sa MSDE i RR u simulacijama sa HPC2N realnim skupom podataka*

Broj poslova	**MSDE**	**RR**
100	+1.52%	+21.01%
200	−3.62%	+15.09%
300	+2.28%	+10.60%
400	+0.11%	+17.68%
500	+3.07%	+18.02%
600	+1.33%	+17.68%
1000	−0.07%	17.87%
2000	+1.38%	+20.34%

Na kraju, kako bi se uzelo u razmatranje balanisranje opterećenja, u simulacijama sa skupom podataka preuzetog iz realnog klaud okruženja, uključen je i DI indikator koji je formulisan prema jednačini (25). Komparativna analiza DI indikatora performansi prikazana je u Tabeli 13 [121]. Najbolji rezultati iz svake kategorije su označeni bold stilom.

Tabela 13: *Komparativna analiza DI indikatora u simulacijama sa HPC2N realnim skupom podataka – MBO-ABC vs. MBO, MSDE, RR*

Broj poslova	RR	MSDE	MBO	MBO-ABC
100	2.015933	0.939974	1.130944	**0.928832**
200	2.132201	1.084389	1.127601	**0.984572**
300	2.116783	**0.979446**	1.259612	1.003461
400	1.990805	**0.839467**	1.102893	0.943278
500	2.135521	0.917298	1.3039980	**0.885800**
600	1.858426	**0.951177**	1.100474	1.015604
1000	2.238352	1.01822	1.365704	**0.969883**
2000	2.085152	**0.971754**	1.152770	0.980129

Potrebno je napomenuti da DI nije uzet u obzir kao cilj optimizacije. Rezultati komparacije indikatora DI su očekivani gde se može zaključiti da je RR heuristika ostvarila najgore vrednosti DI, dok je MBO metaheuristika ostvarila veće DI vrednosti od MSDE i MBO-ABC metaheuristika. Na kraju, MSDE i MBO-ABC ostvaruju približno iste performanse za ovaj indikator.

Kao generalni zaključak, MBO-ABC hibridna metaheuristika može da ostvari bolji kvalitet rešenja, robusnost i skalabilnost u odnosu na druge algoritme koji su uključeni u komparativnoj analizi. Takođe, primenjujući MBO-ABC poboljšanja se mogu uspostaviti u problemu raspoređivanja poslova u klaud okruženju.

7.5 MBO-ABC zaključak

U ovom delu disertacije detaljno je prikazana implementacija hibridnog MBO-ABC algoritma, koji prevazilazi uočene nedostatke osnovne MBO metaheuristike. Algoritam je prvo testiran na standardnom skupu problema globalne optimizacije bez ograničenja, a zatim su izvršeni testovi na modelu raspoređivanja poslova u klaud okruženju sa jednom funkcijom cilja i to sa veštačkim i pravim skupom podataka.

Za sve instance testova, urađena je komparativna analiza sa osnovnim MBO algoritmom i sa drugim vrhunskim pristupima, koji su testirane na istim instancama problema i čiji su rezultati publikovani u vrhunskim međunarodnim časopisima.

Na osnovu rezultata istraživanja, zaključeno je da sledeće:

1. unapređeni MBO postiže bolje performanse od osnovne MBO implementacije, čime je osnovna MBO verzija unapređena i

2. hibridni MBO-ABC algoritam generiše bolja rešenja problema raspoređivanja poslova i balansiranja opterećenja od drugih heurisitka i metaheuristika, čime je Hipoteza X dokazana (pogledati Glavu 1).

Rezultati do kojih se došlo ovim istraživanjem publikovani su međunarodnom časopisu od nacionalnog značaja referisanog u Scopus bazi [121].

8 Implementacija hibridnog WOA algoritma za rešavanje problema raspoređivanja poslova i balansiranja opterećenja u klaud računarstvu

Drugi algoritam koji je implementiran i adaptiran za potrebe istraživanja ove doktroske disertacije predstavlja unapređenu verziju osnovne WOA metaheuristike. Teorijskom analizom i praktičnim simulacijama sa osnovnim WOA algoritmom, ustanovljeni su određeni nedostaci. Unapređena verzija WOA metaheurisike eliminiše nedostatke osnovne verzije tako što je primenjena hibridizacija sa druga dva poznata algoritma inteligencije rojeva. Unapređena WOA implementacija je prvo testirana na širem skupu standardnih benčmark problema globalne optimziacije sa ograničenjima vrednosti parametara, a zatim i na problemu raspoređivanja poslova i balansiranja opterećenja na klaudu.

Detaljan prikaz osnovne WOA implementacije dat je u Poglavlju 6.2.2.

U prvom delu ove glave prikazani su nedostaci originalne WOA metaheuristike, gde su pojedinačno istaknuti i objašnjeni svi elementi koji se razmatraju za dalja unapređenja. Nakon toga, polazeći od uočenih nedostataka, u drugom delu ove glave prikazani su implementacioni detalji hibridnog WOA pristupa, koji neutrališe nedostatke osnovne verzije.

Slično kao i u slučaju verifikovanja unapređenja hibridnog MBO algoritma u odnosu na osnovni MBO, sledeći uobičajenu praksu eksperata iz domena metaheuristika rojeva, kada se kreira novi algoritam, ili kada se unapredi verzija postojećeg algoritma, sa ciljem evaluiranja performansi novog pristupa, prvo se izvode testovi na širem skupu benčmark problema. Zbog toga su u trećem delu ove glave prikazani rezultati simulacija na standardnim problemima globalne optimizacije. Takođe, prikazana je i detaljna komparativna analiza unapređenog WOA algoritma sa osnovnim WOA, kao i sa drugim vrhunskim algoritmima inteligencije rojeva, čiji su rezultati publikovani u svetskoj literaturi.

U četvrtom delu, dati su rezultati testiranja unapređenog hibridnog WOA na praktičan problem raspoređivanja poslova u klaud okruženju, korišćenjem veštačkog skupa podataka (generisanih u okruženju CloudSim simulatora) i realnog skupa podataka, koji su preuzeti iz jedne svetske baze. U oba slučaja, hibridizovani WOA

testiran je na modelu raspoređivanja poslova u klaud okruženju sa više funkcija cilja i ograničenjima (pogledati Poglavlje 4.3). Takođe, u ovom delu disertacije data je detaljna komarativna analiza sa osnovnom WOA metaheuristikom i drugim heuristikama i metaheuristikama koje su testirane na istoj instanci problema.

Konačno, na kraju ove glave dat je kratak zaključak eksperimenata sa hibridnim WOA algoritmom i izvedene su implikacije na osnovne hipoteze koje su formulisane u Glavi 1.

Rezultati do kojih se došlo ovim istraživanjem publikovani su u radu u jednom vrhunskom međunarodnom časopisu kategorije M22 [114].

8.1 Nedostaci osnovne WOA metaheuristike

Izvršavanjem empirijskih testiranja originalnog WOA algoritma, zaključeno je da postoje mogućnost i prostor za određena unapređenja. WOA algoritam je testiran na standardnim benčmark funkcijama bezuslovne optimizacije sa ograničenjima vrednosti promenljivih, gde su primećena dva nedostatka: neprilagođeni balans eksploatacije i eksploracije i preuranjena konvergencija.

Proces eksploracije u originalnom WOA pristupu se izvršava samo u slučaju kada su oba uslova $p < 0.5$ i $|A \geq 1|$ zadovoljena. Osim toga, WOA mehanizam eksploracije usmerava proces pretrage prema postojećim slučajnim rešenjima iz populacije i zatim izvršava pretragu u bliskom okruženju takvih rešenja (pogledati jednačinu (64). Nova pseudo-slučajna rešenja koja se nalaze u domenu prostora pretrage se uopšte ne razmatraju.

U svim drugim slučajevima, proces pretrage originalnog WOA izvršava intenzifikaciju tako što generiše nova rešenja u pravcu trenutno najboljeg vektora rešenja $\vec{X}^*$. Ako su uslovi $p < 0.5$ i $|A < 1|$ zadovoljeni, proces pretrage se izvršava tako što se koriste jednačine (57) - (60), a kada je uslov $p >= 0.5$ ispunjen, koristi se jednačina (62).

Ovakvo ponašanje WOA metaheuristike manifestuje neusklađenost balansa između intenzifikacije i diversifikacije u ranijim iteracijama izvršavanja algoritma. U ovim iteracijama, snaga eksploracije u nekim puštanjima nije dovoljna da bi WOA uspeo da pronađe domen prostora pretrage gde je locirano optimalno rešenje. Implikacije ovoga su lošije prosečne vrednosti i kvalitet rešenja.

Takođe, na osnovu rezultata empirijskih simulacija, zaključeno je da balans između eksploatacije i eksploracije, koji je pomeren na stranu eksploatacije može da dovede do preuranjene konvergencije, kada se proces pretrage zaglavi u nekom od suboptimalnih regiona. Preuranjena konvergencija se javlja kada dva interna WOA kontrolna parametra $\vec{A}$ i $\vec{C}$, ne mogu da generišu bolja rešenja u uzastopnim iteracijama i populacija gubi diverzitet (eng. population diversity). Konačno, u izvršavanjima algoritma kada dođe do preuranjene konvergencije, WOA ne pokazuje zadovoljavajući brzinu konvergencije.

Tako na primer, u nekim puštanjima algoritma, pod uticajem faktora slučajnosti, početna populacija se generiše dosta dalje od optimalnog domena prostora pretrage. Izvršavanjem procedure eksploracije, koja istražuje blisko okruženje početnih rešenja, proces pretrage ne može da konvergira prema optimalnom regionu. U takvim slučajevima, rešenja ne mogu da se poboljšaju u u uzastopnim iteracijama, i cela populacija se sastoji iz relativno sličnih rešenja (razlika može biti samo u nekoliko komponenti rešenja) koja su daleko od optmalnog regiona.

Pretragom svetske literature vidi se da su neki autori predložili unapređene/hibridizovane verzije WOA metaheuristike, koje eliminišu neke od nedostataka osnovne verzije. Istraživači Mafarja i Mirjalili prikazali su dve hibridizovane WAO metaheuristike, koje koriste mehanizme nasleđene iz algoritma SA, kako bi se poboljšalo globalno najbolje rešenje koje generiše WOA standardni proces pretrage [70]. Metaheuristika LWOA (eng. Lévy fight trajectory-based whale optimization algorithm) funkcioniše na način da, nakon što se rešenja ažuriraju WOA procesom pretrage, drugo ažuriranje se izvršava pomoću putanja Lévy leta (eng. Levy fight trajectory) [68]. U [56], unapređen WOA za rešavanje skeletalnih strukturnih problema je predložen. Unapređeni algoritam u svakoj iteraciji ažurira samo odabrane parametre izabranih kandidat rešenja. Istraživači u [18], su kreirali unapređenu verziju WOA algoritma koji primenjuje oprator mutacije DE pristupa i koji koristi adaptivnu strategiju balansiranja eksploatacije i eksploracije.

8.2 Prikaz hibridizovanog WOA algoritma

Uzimajući u obzir da su dva najbitnija mehanizma svake metaheuristike rojeva eksploatacija (intenzifikacija) i eksploracija (diverzifikacija), kao i da se efikasnost

algoritma inteligencije rojeva sagledava u odnosu na brzinu konvergencije i kvalitet rešenja, koji zavise od podešavanja balansa između ova dva procesa, osnova za unapređenje originalnog WOA algoritma bila je balans između eksploatacije i eksploracije.

Iz opšte perspektive, bilo koji algoritam inteligencije rojeva može da se unapredi primenom manjih i/ili većih strategija modifikacije. Manja unapređenja obuhvataju modifikacije nekih komponenti jednačine/jednačina pretrage, kao i podešavanje ponašanja i vrednosti kontrolnih parametara algoritma. Veća unapređenja se uobičajeno odnose na postupak hibridizacije sa drugim metaheuristikama ili heuristikama. Hibridni algoritmi kombinuju najbolje sposobnosti (karakteristike) dva ili više pristupa, tako što zamenjuju slabosti jednog prednostima nekog drugog metoda ili algoritma.

Istraživanjem dostupne literature iz domena računarskih nauka, a na osnovu analize rezultata u publikovanim radovima, može da se zaključi da hibridni algoritmi mogu da budu veoma efikasni u rešavanju različitih tipova problema [97], [122], [37]. Dokazano je da hibridni pristupi značajno mogu da unaprede originalne implementacije [136], [135], [141], [116], [121].

Takođe, u literaturi mogu da se nađu i hibridi koji kombinuju druge metode veštačke inteligencije sa algoritmima rojeva. Kao primer navode se veštačke neuronske mreže (eng. artificial neural networks - ANNs) i konvolutivne neuronske mreže (eng. convolutional neural networks - CNNs) koje efikasno mogu da se kombinuju sa algoritmima inteligencije rojeva [65], [111].

Modifikacije hibridizovanog WOA pristupa u odnosu na osnovni WOA mogu da se sumiraju na sledeći način:

- prvo, usvojen je mehanizam eksploracije iz ABC metaheuristike;
- drugo, uveden je dodatni dinamički kontrolni parametar koji kontroliše novi mehanizam eksploracije i
- treće, predloženi pristup uključuje jednačinu pretrage iz metaheuristike pretrage svitaca (FA).

Uzimajući u obzir sve navedene modifikacije, predloženi pristup je nazvan WOA ABC eksploracija pretraga svitaca (eng. WOA ABC exploration firefly search WOA-AEFS).

U opisu algoritma korišćene su sledeće notacije: svako kandidat rešenje i iz populacije se prikazuje kao vektor $\vec{X}_i$, gde se svako rešenje sastoji iz M komponenti (promenljivih odlučivanja) $\vec{X}_i = x_i^1, x_i^2, ...x_i^M$. Vrednost svake promenljive odlučivanja j, i-tog rešenja (x_i^j) uzima se iz domena $[lb_j, ub_j]$, gde lb_j i ub_j označavaju donju i gornju granicu prostora pretrage u j-toj dimenziji, respektivno. Notacijom $fit(\vec{X}_i)$ označava se podobnost i-tog rešenja, dok je vrednost funkcije cilja istog rešenja označena kao $f(\vec{X}_i)$.

8.2.1 ABC mehanizam eksploracije i dodatni kontrolni parametri

Kao što je i navedno u Poglavlju 8.1, u originalnoj WOA implementaciji, proces eksploracije se vrši samo u scenariju kada su oba uslova $p < 0.5$ i $|A \geq 1|$, zadovoljena. Takođe, na osnovu jednačine (64), vidi se da je proces eksploracije orijentisan samo u smeru postojećih rešenja iz populacije i nova, još uvek neistražena rešenja se ne generišu.

Kao posledica ovoga, u nekim puštanjima algoritma, kada se kreirana rešenja nalaze dalje od optimalnog domena prostora pretrage, eksploracija oko bliskog okruženja postojećih rešenja nije dovoljna da bi proces pretrage konvergirao ka optimalnom domenu. S druge strane, mehanizam eksploracije ABC algoritma generiše kompletno nasumična rešenja, na isti način kao i u fazi inicijalizacije algoritma, kada se inicijalizuje slučajna početna populacija. Navedno je bilo osnovna motivacija za pokušaj unapređenja osnovne WOA metaheuristike hibridizacijom sa ABC pristupom [114].

Za svako rešenje $\vec{X}_i$ iz populacije, uključena je dodatna varijabla $trial_i$. U svakoj iteraciji, kada rešenje $\vec{X}_i$, ne može da se poboljša, njegova $trial_i$ vrednost se povećava za 1. Kada $trial$ vrednost određenog rešenja dostigne predetermisani prag tolerancije $limit$, to rešenje se izbacuje iz populacije i zamenjuje se drugim, nasumično generisanim rešenjem u okviru donjih i gornjih granica prostora pretrage, pri čemo se koristi sledeća jednačina:

$$x_i^j = \theta \cdot (ub_j - lb_j) + lb_j, \tag{68}$$

gde je x_i^j, j-ta promenljiva odlučivanja i-tog rešenja, a θ označava pseudo-slučajan broj iz opsega $[0, 1]$.

Ovakav mehanizam eksploracije, pogotovo u ranim iteracijama izvršavanja algoritma, značajno poboljšava brzinu konvergencije i omogućava zadržavanje diverziteta čitave populacije. Čak i u slučajevima kada se pretraga zaglavi u nekom od suboptimalnih regiona, ovakav mehanizam je u stanju da efikasno usmeri proces pretrage prema nekim drugim domenima regiona koji se pretražuje. Međutim, u kasnijim iteracijama, kada proces pretrage konvergira ka optimalnom regionu i kada je potrebna fino podešena (usmerena) pretraga, primonom mehanizma ABC eksploracije potencijalno dobra rešenja, koja nisu poboljšana u nekoliko poslednjih iteracija mogu da budu izgubljena [114].

Da bi se ovakav scenario prevazišao, uključen je dodatni kontrolni parametar, stopa uticaja eksploracije (eng. exploration influence rate) *eir*, čija se vrednost izražava u procentima.

Na početku puštanja algoritma, vrednost *eir* je podešena na 100, što znači da je cela populacija (100% populacije) pod uticajem ABC eksploracije. Međutim, tokom jednog puštanja algoritma, vrednost *eir* se postepeno smanjuje, dok ne dostigne graničnu vrednost (prag), koja je u ovoj implementaciji podešena na 5. Na primer, ako je vrednost *eir* 50, to znači da su 50% najlošijih kandidat rešenja izložena riziku da budu odbačeni iz populacije ako je njihova *trial* vrednost dostigla predefinisanu *limit* vrednost. Sa ciljem uspešne primene opisanog mehanizma, pre nego što se mehanizam ABC eksploracije primeni, cela populacija se sorira na osnovu vrednosti kriterijuma podobnosti u opadajućem redosledu [114].

Na osnovu izvršenih empirijskih eksperimenata, došlo se do zaključka da približno optimalna vrednost dimamičkog *eir* parametra može da se izračuna kao [114]:

$$eir = 100 \cdot (1 - \frac{t}{maxIter}) \tag{69}$$

8.2.2 FA jednačina pretrage

Sledeći cilj daljeg unapređenja originalne WOA metaheuristike, poboljšavanjem brzine konvergencije, inkorporirana je strategija pretrage svitaca u hibridizovanom WOA-AEFS pristupu. Na osnovu istraživanja u [135], FA jednačina pretrage može značajno da unapredi brzinu konvergencije. Kao što je i navedeno u Poglavlju 6.3, FA metaheuristiku je kreirao Yang 2009. godine [152].

U svakoj iteraciji izvršavanja hibridnog WOA-AEFS pristupa, ako je uslov $p_1 \geq 0.5$ zadovoljen, proces eksploatacije može da se izvrši korišćenjem WOA mehanizma spiralne strategije (jednačina 62) ili primenom FA jednačine pretrage sa jednakom verovatnoćom p_1. Strategija pretrage FA se primenjuje za svaki parametar j rešenja i korišćenjem sledeće jednačine [114]:

$$x_i^j(t+1) = x_i^j + \beta_0 \cdot e^{-\gamma * r_{i,k}^2}(x_k^j - x_i^j) + \alpha * (rand - 0.5), \tag{70}$$

gde β_0 označava privlačenje svitaca na razdaljini $r = 0$, varijacija privlačenja je definisana γ parametrom, α je parametar slučajnosti (eng. randomization parameter) FA metaheuristike, $rand$ je pseudo-slučajan broj između 0 i 1, dok k predstavlja slučajno rešenje iz populacije.

Razdaljina između rešenja i i k, označena kao $r_{i,k}$, računa se kao Dekartova razdaljina [152]:

$$r_{i,k} = ||\vec{X}_i - \vec{X}_k|| = \sqrt{\sum_{j=1}^{M}(x_{i,j} - x_{k,j})^2}, \tag{71}$$

gde je M broj promenljivih odlučivanja (komponenti rešenja).

Kao i u originalnom FA pristupu, u ovoj implementaciji je korišćena dinamička vrednost za parametar $alpha$ koja zavisi od trenutne iteracije t i najvećeg broja iteracija ($maxIter$) u jednom izvršavanju algoritma [152]:

$$\alpha(t) = (1 - (1 - ((10^{-4}/9)^{1/maxIter}))) \cdot \alpha(t-1) \tag{72}$$

Za više informacija o FA kontrolnim parametrima, pogledati rad [152].

8.2.3 Inicijalizacija populacije i WOA-AEFS pseudo kod

Kao u svakom pristupu inteligencije rojeva, početna populacija se inicijalizuje na početku svakog puštanja algoritma. U WOA-AEFS fazi inicijalizacije, populacija od N rešenja $\vec{X}_i(i = 1, 2, 3, ...N)$, gde se svako rešenje sastoji iz M promenljivih odlučivanja $\vec{X}_i = x_i^1, x_i^2, ...x_i^M$, se generiše. Cela populacija se označava matricom veličine NxM:

$$P = \begin{bmatrix} x_1^1 & x_1^2 & x_1^3 & \cdots & x_1^j & \cdots & x_1^M \\ x_2^1 & x_2^2 & x_2^3 & \cdots & x_2^j & \cdots & x_2^M \\ x_3^1 & x_3^2 & x_3^3 & \cdots & x_3^j & \cdots & x_3^M \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ x_N^1 & x_N^2 & x_N^3 & \cdots & x_N^j & \cdots & x_N^M \end{bmatrix}.$$

Svaki parametar (promenljiva odlučivanja) j svakog rešenja u populaci se generiše u fazi inicijalizacije primenom jednačine (68). Osnovni koraci izvršavanja predloženog WOA-AEFS algoritma su prikazani u Algoritmu 7 [114].

Algoritam 7 Pseudo kod WOA-AEFS

Inicijalizacija. Generisanje slučajne početne populacije P korišćenjem jednačine (68); podešavanje vrednosti *trial* parametra svih rešenja na 0.
Inicijalizacija kontrolnih parametara. Inicijalizacija vrednosti za ABC kontrolne parametre eksploracije *eir* i *limit* i FA parametar pretrage α.
Izračunavanje podobnosti. Izračunavanje podobnost svakog kandidat rešenja iz populacije i određivanje trenutno najbolje globalno rešenje $\vec{X}^*$
while $(t < maxIter)$ **do**
  for svako kandidat rešenje **do**
    Ažuriranje vrednosti za A, $\vec{C}$, $\vec{a}$, l, p, p_1
    if $p < 0.5$ **then**
      if |A|<1 **then**
        Ažuriranje pozicija kandidat rešenja $\vec{X}$ korišćenjem jednačine (58) i čuvanje pozicija novog rešenja u vektor $\vec{X}_{new}$
      else if $|A \geq 1|$ **then**
        Selekcija slučajnog rešenja $\vec{X}_{rnd}$ iz populacije
        Ažuriranje pozicija kandidat rešenja $\vec{X}$ korišćenjem jednačine (64) i čuvanje pozicija novog rešenja u vektor $\vec{X}_{new}$
      end if
    else if $p \geq 0.5$ **then**
      if $p_1 < 0.5$ **then**
        Ažuriranje pozicija kandidat rešenja $\vec{X}$ korišćenjem FA jednačine pretrage (jednačina (70)) i čuvanje pozicija novog rešenja u $\vec{X}_{new}$
      else if $p_1 \geq 0.5$ **then**
        Ažuriranje pozicija kandidat rešenja $\vec{X}$ primenom WOA spiralne pretrage (jednačina (62)) i čuvanje pozicija novog rešenja u vektor $\vec{X}_{new}$
      end if
    end if
    Izbor između starog $\vec{X}$ i novog $\vec{X}_{new}$ rešenja primenom mehanizma pohlepne selekcije
    Ako je novo rešenje izabrano, zameniti $\vec{X}$ sa $\vec{X}_{new}$ i podesiti *trial* vrednost na 0.
    Ako je staro rešenje izabrano, povećati *trial* vrednost rešenja $\vec{X}$.
  end for
  Svako rešenje čije komponente imaju vrednosti van dopustivog regiona domena pretrage, vratiti u okvire dozvoljenih granica
  Evaluiranje svih rešenja iz populacije i računanje podobnosti
  Sortiranje svih rešenja po osnovu kriterijuma podobnosti u opadajućem redosledu

  Zamena svih rešenja koja pripadaju eif % najgorih rešenja u populaciji i za koje je uslov $trial \geq limit$ zadovoljen sa slučajnim rešenjem generisanim pomoću jednačine (68).
  Ažuriranje pozicija globalno najboljeg rešenja $\vec{X}^*$ ako je potrebno
  $t = t + 1$
  Izračunavanje novih vrednosti dinamičkih parametara *eir* i α pomoću izraza (69) i (72), respektivno.
end while
return Najbolje rešenje $(\vec{X}^*)$ iz populacije

8.3 Simulacije na standardnim benčmark problemima

Kada se kreira nova metaheuristika, ili se postojeća unapređuje, bilo sitnim modifikacijama, bilo hibridizacijom sa drugim pristupima, uobičajena je praksa da se novi pristup prvo testira na širem opsegu benčmark problema, sa ciljem preciznijeg utvrđivanja performansi. Zbog ovog razloga, hibridizovani WOA-AEFS algoritam je prvo testiran na standardnim i poznatim benčmark problemima bez ograničenja. Na ovaj način, prednosti hibridizovane metaheuristike u odnosu na osnovnu implementaciju, uzimajući u obzir osnovne indikatore performansi, brzinu konvergencije i kvaliteta rešenja, mogu kvantitativno da se izraze.

U nastavku su prvo prikazane formulacije standardnih benčmark funkcija koje su korišćene u simulacijama zajedno sa podešavanjima kontrolnih parametara WOA-AEFS metaheuristike, a zatim je prikazana komparativna analiza sa osnovnim WOA pristupom i drugim vrhunskim metaheuristikama koje su testirane sa istim instancama benčmark problema.

8.3.1 Matematičke formulacije standardnih benčmark funkcija

Sa ciljem validiranja robusnosti, kvalitet rešenja i konvergencije WOA-AEFS algoritma, korišćene su 23 klasične benčmark funkcije bez ograničenja. Originalni WOA algoritam je takođe testiran na istim benčmark instancama i rezultati su prikazani u publikovanom radu [80].

Zarad lakše preglednosti, prateći praksu iz svetske naučne literature [153], benčmark setovi su podeljeni na dve grupe: unimodalne i multimodalne. Dalje, iz kategorije multimodalnih test funkcija, posebno su izdvojene funkcije sa fiksnim brojem parametara. Osnovna razlika između klasičnih multimodalnih i multimodalnih funkcija fiksnih dimenzija je mogućnost određivanja broja parametara (komponenti). U slučaju benčmark multimodalnih funkcija fiksnih dimenzija, u eksperimentalnim simulacijama, broj parametara ne može da se podešava.

Za svaku funkciju (unimodalnu i multimodalnu), jedinstveni identifikator (ID) je dodeljen, pre svega zbog lakše klasifikacije rezultata i bolje čitljivosti. U Tabeli 14 prikazane su osnovne karakteristike unimodalnih benčmark funkcija koje su korišćene u simulacijama, dok su matematičke formulacije date u Tabeli 15.

Tabela 14: *Osnovne Karakteristike unimodalnih benčmark funkcija*

ID	Naziv benčmarka	Separabilnost	Konveksnost	Skalabilnost
F1	Sfera	Separabilno	Konveksno	Skalabilno
F2	Schwefel Problem 2.22	Neseparabilno	Konveksno	Skalabilno
F3	Schwefel Problem 1.2	Separabilno	Konveksno	Skalabilno
F4	Schwefel Problem 2.21	Separabilno	Konveksno	Skalabilno
F5	Generalizovana Rosenbrock Funkcija	Ne-separabilno	Ne-konveksno	Skalabilno
F6	Step 2 Funkcija	Separabilno	Konveksno	Skalabilno
F7	Quartic Funkcija sa šumom	Separabilno	Nekonveksno	Skalabilno

Tabela 15: *Matematičke formulacije unimodalnih benčmark funkcija*

ID	Formulacija	Br. Dim.	Opseg	Globalni Minimum
F1	$f(x) = \sum_{i=1}^{n} x_i^2$	30	$[-100,100]$	$f(x^*) = 0$ at $x^* = (0, ..., 0)$
F2	$f(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	$[-10,10]$	$f(x^*) = 0$ at $x^* = (0, ..., 0)$
F3	$f(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	30	$[-100,100]$	$f(x^*) = 0$ at $x^* = (0, ..., 0)$
F4	$f(x) = \max_i \{\lvert x_i \rvert, 1 \leq i \leq n\}$	30	$[-100,100]$	$f(x^*) = 0$ at $x^* = (0, ..., 0)$
F5	$f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	$[-30,30]$	$f(x^*) = 0$ at $x^* = (1, ..., 1)$
F6	$f(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$	30	$[-100,100]$	$f(x^*) = 0$ at $x^* = (0, ..., 0)$
F7	$f(x) = \sum_{i=1}^{n} i x_i^4 + rand[0, 1)$	30	$[-1.28,1.28]$	$f(x^*) = 0$ at $x^* = (0, ..., 0)$

Osnovne karakteristike multimodalnih benčmark funkcija, koje su korišćene u simulacijama date su u Tabeli 16, dok su njihove matematičke formulacije prikazane u Tabeli 17. Funkcije u Tabeli 16 sa ID-jem u rasponu $F8 - F13$ pripadaju kategoriji klasičnih multimodalnih benčmark funkcija, dok funkcije sa ID-jem iz opsega $F14 - F23$ pripadaju grupi multimodalnih funkcija fiksnih dimenzija.

Tabela 16: *Karakteristike multimodalnih benčmark funkcija*

ID	Naziv benčmarka	Separabilnost	Konveksnost	Skalabilnost
F8	Generalizovani Schwefel Problem 2.26	Separabilno	Konveksno	Skalabilno
F9	Generalizovana Rastrigin Funkcija	Ne-separabilno	Konveksno	Skalabilno
F10	Ackley Funkcija	Ne-separabilno	Ne-konveksno	Skalabilno
F11	Generalizovana Griewank Funkcija	Ne-separabilno	Ne-konveksno	Skalabilno
F12	Generalizovana Penalized Funkcija	Ne-separabilno	Ne-konveksno	Skalabilno
F13	Generalizovana Penalized Funkcija	Ne-separabilno	Ne-konveksno	Skalabilno
F14	Shekel's Foxholes Funkcija	Ne-separabilno	Ne-konveksno	Skalabilno
F15	Kowalik Funkcija	Ne-separabilno	Ne-konveksno	Skalabilno
F16	Six-Hump Camel-Back Funkcija	Ne-separabilno	Ne-konveksno	Ne-skalabilno
F17	Branin Funkcija	Ne-separabilno	Ne-konveksno	Ne-skalabilno
F18	Goldstein-Price Funkcija	Ne-separabilno	Ne-konveksno	Ne-skalabilno
F19	Hartman's Family (Hartman 3 Funkcija)	Ne-separabilno	Ne-konveksno	Ne-skalabilno
F20	Hartman's Family (Hartman 6 Funkcija)	Ne-separabilno	Ne-konveksno	Ne-skalabilno
F21	Shekel's Family (Shekel 5 Funkcija)	Ne-separabilno	Ne-konveksno	Skalabilno
F22	Shekel's Family (Shekel 7 Funkcija)	Ne-separabilno	Ne-konveksno	Skalabilno
F23	Shekel's Family (Shekel 10 Funkcija)	Ne-separabilno	Ne-konveksno	Skalabilno

Tabela 17: *Matematičke formulacije multimodalnih benčmark funkcija*

ID	Formulacija	Br. Dim.	Opseg	Globalni Minimum
F8	$f(x) = \sum_{i=1}^{n} [-x_i \sin(\sqrt{\lvert x_i \rvert})]$	30	$[-500,500]$	$f(x^*) = -418.9829 \times 5$
F9	$f(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	$[-5.12,5.12]$	$f(x^*) = 0$ at $x^* = (0, ..., 0)$
F10	$f(x) = -20e^{(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2})} - e^{(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i))} + 20 + e$	30	$[-32,32]$	$f(x^*) = 0$ at $x^* = (0, ..., 0)$
F11	$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$	30	$[-600,600]$	$f(x^*) = 0$ at $x^* = (0, ..., 0)$
F12	$f(x) = \frac{\pi}{30}[10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2] + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{1}{4}(x_i + 1)$	30	$[-50,50]$	$f(x^*) = 0$ at $x^* = (1, ..., 1)$
F13	$f(x) = 0.1[\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]] + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30	$[-50,50]$	$f(x^*) = 0$ at $x^* = (1, ..., 1)$
F14	$f(x) = [\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{i,j})^6}]^{-1}$	2	$[-65.536,65.536]$	$f(x^*) = 0.99$ at $x^* \approx -31.97$
F15	$f(x) = \sum_{i=0}^{10} [a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}]^2$	4	$[-5,5]$	$f(x^*) = 0.00030$ at $x^* = [0.1928, 0.1908, 0.1231, 0.1357]$
F16	$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$[-5,5]$	$f(x^*) = -1.03$ at $x^* = \approx (\pm 0.08, \pm 0.71)$
F17	$f(x) = \left(x_2 - \frac{5.1x_1^2}{4\pi^2} + \frac{5x_1}{\pi} - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$	2	$x_1 = [-5,10]$, $x_2 = [0,15]$	$f(x^*) = 0.39$ at $x^* = \approx (\pm 0.08, \pm 0.71)$
F18	$f(x) = [1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]$ x$[30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	$[-2,2]$	$f(x^*) = 3$ at $x^* = (0, -1)$
F19	$f(x) = -\sum_{i=1}^{4} c_i \, e^{\left[-\sum_{j=1}^{3} a_{i,j}(x_j - p_{i,j})^2\right]}$	3	$[0,1]$	$f(x^*) = -3.86$ at $x^* \cong (0.1, 0.55, 0.85)$
F20	$f(x) = -\sum_{i=1}^{4} c_i \, e^{\left[-\sum_{j=1}^{6} a_{i,j}(x_j - p_{i,j})^2\right]}$	6	$[0,1]$	$f(x^*) = -3.32$ at $x^* \cong (0.20, 0.15, 0.47, 0.27, 0.31, 0.65)$
F21	$f(x) = -\sum_{i=1}^{5} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[0,10]$	$f(x^*) = -10.1532$ at $x^* = (4, 4, 4, 4)$
F22	$f(x) = -\sum_{i=1}^{7} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[0,10]$	$f(x^*) = -10.4028$ at $x^* = (4, 4, 4, 4)$
F23	$f(x) = -\sum_{i=1}^{10} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[0,10]$	$f(x^*) = -10.5363$ at $x^* = (4, 4, 4, 4)$

U simulacijama sa test funkcijama $F1 - F13$ korišćeno je 30 dimenzija ($D = 30$), kao i u [80]. Broj dimenzija u simulacijama sa multimodalnim funkcijama fiksnih dimenzija može se videti iz Tabele 17.

8.3.2 Podešavanje kontrolnh parametara hibridnog WOA-AEFS algoritma

Da bi se izvršila objektvina evaluacija unapređenja performansi WOA-AEFS algoritma u odnosu na originalni WOA, u svim testovima je korišćena populacija sa 30 individua ($N = 30$), čiji se kvalitet inkremenalno poboljšavan tokom 500 iteracija ($maxIter = 500$) u svakom puštanju algoritma. Ovakvih podešavanjima generiše se ukupan broj od 15,000 evaluacija funkcije cilja ($30x500 = 15000$). Iste vrednosti su korišćene i u [80].

Obzirom na to da WOA-AEFS, osim što koristi opšte parametre optimizacionih algoritama (N i $maxIter$), koristi i WOA, ABC i FA specifične parametre pretrage, radi bolje preglednosti, svi kontrolni parametri su podeljeni u četiri grupe: opšti parametri, WOA parametri, parametri mehanizma ABC eksploracije i FA parametri pretrage.

Podešavanja kontrolnih parametara, kao i ponašanje dinamičkih parametara tokom jednog izvršavanja algoritma, sumirano je u Tabeli 18 [121].

Takođe treba da se naglasi da su uzete iste vrednosti za specifične parametre FA pretrage kao u [152]. S druge strane, na osnovu ranije sprovedenih istraživanja, ABC eksploracija je najefikasnija, ako vrednost parametra $limit$ zavisi od vrednosti N i $maxIter$ [50], [31], [12], [8]. Zbog toga je vrednost parametra $limit$ izračunata kao:

$$limit = round(\frac{maxIter}{N}), \tag{73}$$

gde funkcija $round()$ zaokružuje argumente na najbližu vrednost celog broja.

Tabela 18: *Vrednosti kontrolnih parametara WOA-AEFS metaheuristike*

Naziv parametra	Vrednost
WOA-AEFS opšti parametri	
Veličina populacije (N)	30
Broj iteracija za svako puštanje algoritma ($maxIter$)	500
WOA parametri pretrage	
Početna vrednost parametra a	2.0
Dinamičko ponašanje parametra a	na osnovu jednačine (61)
ABC parametri eksploracije	
Parameter *limit*	17
Početna vrednost parametra *eir*	100
Dinamičko ponašanje parametra *eir*	na osnovu jednačine (69)
FA parametri pretrage	
Početna vrednost parametra nasumičnosti α	0.5
Dinamičko ponašanje parametra α	na osnovu jednačine (72)
Privlačnost na razdaljini r=0 β_0	0.2
Koeficijent absorpcije γ	1.0

8.3.3 Komparativna analiza i diskusija

Algoritam WOA-AEFS evaluiran je u svim simulacijama u 100 nezavisnih puštanja. U svakom puštanju algoritma inicijalizovana je nasumična populacija veličine N korišćenjem jednačine (68). Za potrebe eksperimenata kreiran je generator pseudo-slučajnih brojeva, i u svakom puštanju algoritma inicijalizovano je drugačije seme pseudo-slučajne sekvence.

Implementacija WOA-AEFS metaheuristike rađena je Javi sa Java SE Development Kit 11 tehnologijom i IntelliJ integrisanim razvojnim okruženjem (eng. integrated development environment - IDE).

U prvim simulacijama, performanse WOA-AEFS hibrida su upoređivane sa performansama originalne WOA metaheuristike. Oba algoritma su testirana na istim benčmark instancama (F1 - F23) i pod istim eksperimentalnim uslovima. Rezultati osnovnog WOA preuzeti su iz [80].

Bez obzira što je osnovni WOA već ranije implementiran i testiran na istim benčmark problemima, tokom sprovedenog istraživanja, implementirana je originalna WOA verzija u Javi, i ostvareni su isti rezultati kao što je prikazano u [80]. Implementacija WOA metaheuristike koja je prikazana u [80] rađena je u okruženju MATLAB alata.

Takođe, u prikazanom simulacijama, korišćeni su isti indikatori performansi kao i u [80] - u 100 nezavisnih puštanja algoritma prikazani su indikatori srednje vrednosti i standardne devijacije (std). U [80], srednje vrednosti su računate za 30 nezavisnih puštanja metaheuristike. Ako se srednje vrednosti računaju na osnovu više putanja algoritma, rezultati su precizniji i faktor slučajnosti ima manji uticaj.

Da bi se odredila unapređenja koja generiše WOA-AEFS u odnosu na osnovnu WOA implementaciju, prvo je urađena komparativna analiza između WOA-AEFS i originalnog WOA, koja je data u Tabeli 19. Sa ciljem bolje vizualizacije postignutih rezultata, bolji rezultati za svaku instancu testa i za svaki pokazatelj performansi su označeni bold stilom.

Kada se unapređuje algoritam (metaheuristika), uvek postoji neki kompromis. Na primer, za jednu benčmark instancu, rezultati se unaprede, dok u slučaju nekih drugih testova, rezultati se pogoršaju. Međutim, bitan je prosek (kada se uzimaju u obzir sve benčmark instance) i tada, u većini implementacija koje su poznate u svetskoj literaturi, unapređena/hibridizovana verzija postiže bolje rezultate od osnovne.

Rezultati prikazani u Tabeli 19 [114] pružaju validan dokaz o unapređenjima originalnog WOA. Benčmark instance i indikatori gde originalni WOA postiže bolje performanse nego WOA-AEFS uključuju sledeće: srednje vrednosti za $F7$ test, std pokazatelj za $F16$ benčmark i vrednosti oba indikatora (srednja vrednost i std) u slučaju testa $F19$. Simulacije, gde oba pristupa ostvaruju iste rezultate obuhvataju vrednosti indikatora srednja vrednost za $F9$, $F15$, $F16$ i $F18$ benčmark instance i std vrednost za test funkciju $F9$.

U svim drugim slučajevima, WOA-AEFS je značajno nadmašio osnovnu WOA metaheuristiku. U Tabeli 19 [114], uključen je dodatan red, gde je izračunato za svaku kolonu, koliko je puta određeni algoritam generisao bolje rezultate. Za 18 od 23 test problema, WOA-AEFS za oba indikatora (prosečne vrednosti i std) je ostvario bolju brzinu konvergencije i kvalitet rezultata u odnosu na originalan WOA.

U osnovnom WOA pristupu, zbog dinamičkog ponašanja parametra $\vec{a}$ (mehanizam prilagođavanja), proces pretrage se ubrzava sa progresom interacija [80]. Ovaj mehanizam se izvršava dobro u slučajevima kada u ranijim iteracijama algoritam uspe da pronađe obećavajući region prostora pretrage. Međutim, ukoliko to nije

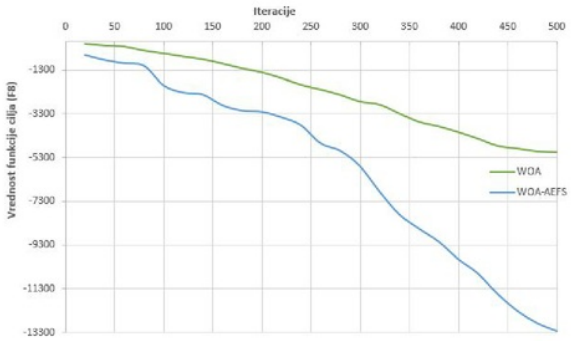
slučaj, algoritam može da se zaglavi u nekom od suboptimalnih regiona, i cela populacija konvergira ka ovom suboptimalnom domenu i gubi diverzitet, tj. algoritam ispoljava ponašanje preuranjene konvergencije. Preuranjena konvergencija je posebno naglašena u $F8$ i $F21$ simulacijama, što je takođe primećeno i u [80]. Glavni razlog preuranjene konvergencije je neadekvatan kompromis između intenzifikacije diverzifikacije, pogotovo u ranim iteracijama izvršavanja algoritma.

Tabela 19: *Komparativna analiza - WOA-AEFS vs. WOA u simulacijama sa benčmark funkcijama bez ograničenja*

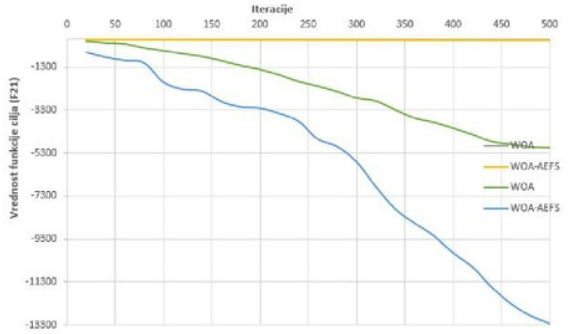
ID	**WOA**		**WOA-AEFS**	
	Prosečna vrednost	*std*	*Prosečna vrednost*	*std*
F1	1.41E-30	4.9E-30	**2.55E-31**	**8.75E-32**
F2	1.06E-21	2.39E-21	**3.97E-23**	**2.99E-23**
F3	5.39E-07	2.93E-06	**6.72E-10**	**5.04E-10**
F4	0.072581	0.39747	**0**	**0**
F5	27.865580	0.763626	**3.29**	**0.012500**
F6	3.116266	0.532429	**1.5E-19**	**7.59E-20**
F7	**0.001425**	0.001149	0.001452	**0.000851**
F8	-5080.76	695.7968	**-13239.3**	**48.3**
F9	0	0	0	0
F10	7.4043	9.897572	**0.013**	**0.0051**
F11	0.000289	0.001586	**0**	**0**
F12	0.339676	0.214864	**9.29E-15**	**2.02E-15**
F13	1.889015	0.266088	**7.56E-14**	**2.09E-14**
F14	2.111973	2.498594	**0.998023**	**1.99E-16**
F15	0.000572	0.000324	**0.000322**	0.000324
F16	-1.03163	**4.2E-07**	-1.03163	2.87E-06
F17	0.397914	2.7E-05	**0.397887**	**1.83E-10**
F18	3	4.22E-15	3	**2.35E-15**
F19	**-3.85616**	**0.002706**	-3.85529	0.003812
F20	-2.98105	0.376653	**-3.31952**	**0.025821**
F21	-7.04918	3.629551	**-10.1532**	**3.02E-12**
F22	-8.181780	3.829202	**-8.925003**	**3.352013**
F23	-9.34238	2.414737	**-10.5364**	**3.5E-13**
Bolji	2	2	18	18

Korišćenjem mehanizma ABC eksploracije, predloženi WOA-AEFS algoritam izbegava preuranjenu konvergenciju, što je empirijski dokazano. U slučajevima $F8$ i $F21$ benčmarka, WOA-AEFS uspešno uspeva da izbegne zaglavljivanje u regionima lokalnog optimuma. Međutim, sa ABC eksploracijom, neka dobra rešenja mogu da se odbace i to WOA-AEFS kompenzuje tako što koristi dinamički parametar *eir* i veoma

efikasnu FA jednačinu pretrage. Takođe, WOA-AEFS još uvek koristi mehanizam adaptivnog skupljanja (parametar $\vec{a}$) originalnog WOA i proces pretrage se time ubrzava, kako algoritam ulazi u kasnije faze izvršavanja. Grafovi brzine konvergencije WOA i WOA-AEFS za $F8$ i $F21$ benčmark instance su dati na slikama 20 i 21, respektivno [114].



Slika 20: *Graf brzine konvergencije WOA-AEFS vs. WOA: F8 benčmark*



Slika 21: *Graf brzine konvergencije WOA-AEFS vs. WOA: F21 benčmark*

Na osnovu analize grafikona prikazanih na slikama 20 i 21, mogu da se izvedu interesantna zapažanja o ponašanju algoritma. Prvo je primećeno da oba algoritma brže konvergiraju sa rastom broja iteracija zbog mehanizma adaptivnog skupljanja. Drugo, može da se vidi da je WOA-AEFS posle približno 300 i 400 iteracija ostvario vrednosti koje osnovni WOA generiše nakon 500 iteracija, kada se posmatraju $F8$ i $F21$ testovi, respektivno. Takođe može da se primeti da se osnovni WOA u $F21$ simulaciji zaglavljuje negde između 80. i 120., kao i između 200. i 220. iteracije. S druge strane, WOA-AEFS pokazuje nepromenljivu konvergenciju tokom celog izvršavanja.

Kao opšti zaključak, na osnovu empirijskih simulacija i teorijskih analiza, može da se izvede da WOA-AEFS znatno poboljšava osnovnu WOA verziju osvrćući se na njene nedostatke u vidu neadekvatno postavljenog balansa između intenzifikacije i diverzifikacije, čime izbegava preuranjenu konvergenciju. Međutim, kao što je već i naglašeno, uvek mora da postoji neki kompromis. Glavna prednost WOA metaheuristike u odnosu na hibridni WOA-AEFS pristup je u tome što WOA koristi manji broj kontrolnih parametara, čime se postiže lakša kontrola izvršavanja algoritma. Da bi se kontrolisala ABC eksploracija i proces pretrage FA, metaheuristika WOA-AEFS uvodi tri dodatna kontrolna parametra, dva dinamička (eir i α) i jedan statički ($limit$).

Osim komparativne analize sa osnovnim WOA algoritmom, takođe je važno da se vidi kako se WOA-AEFS ponaša u odnosu na druge metaheuristike rojeva za iste benčmark instance. Zbog toga je prikazana i komparativna analiza između WOA-AEFS i algoritma PSO [57], algoritma pretrage na osnovu sile gravitacije (eng. gravitation search algorithm - GSA) [95], DE [103], i pristupa brzog evolutivnog programiranja (eng. fast evolutionary programming - FEP) [149]. Rezultati svih navedenih metaheuristika su uzeti iz [80] i [81].

Isto kao i u prvoj komparativnoj analizi, uzeti su indikatori performansi prosečne vrednosti i standardna devijacija koje su izračunate u 100 nezavisnih puštanja algoritma. Komparativna analiza, u koju je uključen i osnovni WOA, data je u Tabeli 20 [114].

Na osnovu rezultata analize, vidi se da algoritam WOA-AEFS u proseku postiže bolje performanse nego svi ostali pristupi. Za indikator standardne devijacije, WOA-AEFS čak za 10 test problema ostvaruje bolje rezultate, dok u slučaju indikatora prosečnih vrednosti, WOA-AEFS ostvaruje najbolje rezultate za 6 test funkcija. Drugi najbolji algoritam je DE, dok je GSA rangiran kao treći najbolji algoritam u ovoj analizi.

Tabela 20: *Komparativna analiza - WOA-AEFS vs. ostali algoritmi u simulacijama sa benčmark funkcijama bez ograničenja*

ID	PSO		GSA		DE		FEP		WOA		HWOA	
	Prosečna vrednost	*std*	*Prosečna vrednost*	*std*	*Prosečna vrednost*	*std*	*Prosečna vrednost*	*std*	*Prosečna vrednost*	*std*	*Prosečna vrednost*	*std*
F1	0.000136	0.000202	2.53E-16	9.67E-17	8.2E-14	5.90E-14	0.00057	0.00013	1.41E-30	4.9E-30	**2.55E-31**	**8.75E-32**
F2	0.042144	0.045421	0.055655	0.194074	1.5E-09	9.9E-10	0.008100	0.000770	1.06E-21	2.39E-21	**3.97E-23**	**2.99E-23**
F3	70.125620	22.119240	896.5347	318.9559	**6.8E-11**	7.4E-11	0.016	0.014	5.39E-07	2.93E-06	6.72E-10	**5.04E-10**
F4	1.086481	0.317039	7.35487	1.741452	0	0	0.3	0.5	0.072581	0.39747	0	0
F5	96.71832	60.11559	67.54309	62.22534	**0**	**0**	5.06	5.87	27.865580	0.763626	3.29	0.012500
F6	0.000102	8.28E-05	2.5E-16	1.74E-16	0	0	0	0	3.116266	0.532429	1.5E-19	7.59E-20
F7	0.122854	0.044957	0.089441	0.04339	0.00463	0.0012	0.1415	0.3522	**0.001425**	0.001149	0.001452	**0.000851**
F8	-4841.29	1152.814	-2821.07	493.0375	-11080.1	574.7	-12554.5	52.6	-5080.76	695.7968	**-13239.3**	**48.3**
F9	46.70423	11.62938	25.96841	7.470068	69.2	38.8	0.046	0.012	0	0	0	0
F10	0.276015	0.50901	0.062087	0.23628	9.7E-08	4.2E-08	0.018	**0.0021**	7.4043	9.897572	**0.013**	0.0051
F11	0.009215	0.007724	27.70154	5.040343	0	0	0.016	0.022	0.000289	0.001586	0	0
F12	0.006917	0.026301	1.799617	0.95114	**7.9E-15**	8E-15	9.2E-06	3.6E-06	0.339676	0.214864	9.29E-15	**2.02E-15**
F13	0.006675	0.008907	8.899084	7.126241	**5.1E-14**	4.8E-14	0.00016	0.000073	1.889015	0.266088	7.56E-14	**2.09E-14**
F14	3.627168	2.560828	5.89838	3.831299	0.998004	3.3E-16	1.22	0.56	2.111973	2.498594	0.998023	**1.99E-16**
F15	0.000577	0.000222	0.003673	0.001647	4.5E-14	0.00033	0.0005	0.00032	0.000572	0.000324	**0.000322**	0.000324
F16	-1.03163	6.25E-16	-1.03163	**4.88E-16**	-1.03163	3.1E-13	-1.03	4.9E-07	-1.03163	4.2E-07	-1.03163	2.87E-06
F17	0.397887	0	0.397887	0	0.397887	9.9E-09	0.398	1.5E-07	0.397914	2.7E-05	0.397887	**1.83E-10**
F18	3	1.33E-15	3	4.17E-15	3	**2E-15**	3.02	0.11	3	4.22E-15	3	2.35E-15
F19	-3.86278	2.58E-15	-3.86278	**2.29E-15**	N/A	N/A	-3.86	0.000014	-3.85616	0.002706	-3.85529	0.003812
F20	-3.26634	0.060516	-3.31778	**0.023081**	N/A	N/A	-3.27	0.059	-2.98105	0.376653	**-3.31952**	0.025821
F21	-6.8651	3.019644	-5.95512	3.737079	-10.1532	0.0000025	-5.52	1.59	-7.04918	3.629551	-10.1532	**3.02E-12**
F22	-8.45653	3.087094	-9.68447	2.014088	**-10.4029**	**3.9E-07**	-5.53	2.12	-8.181780	3.829202	-8.925003	3.352013
F23	-9.95291	1.782786	-10.5364	**2.6E-15**	-10.5364	1.9E-07	-6.57	3.14	-9.34238	2.414737	-10.5364	3.5E-13
Najbolji	0	0	0	4	4	1	0	1	1	0	6	10

8.4 Simulacije WOA-AEFS metaheuristike za problem raspoređivanja poslova na klaudu

U ovom delu disertacije detaljno su prikazani empirijski rezultati koje je WOA-AEFS algoritam ostvario u rešavanju problema raspoređivanja poslova u klaud okruženju. Da bi se bolje i preciznije evaluirale performanse predloženog pristupa, izvršene su dve grupa simulacija [114]: jedna sa pravim skupom podataka i druga sa veštačkim skupom podataka.

U prvoj grupi simulacija, korićen je model sa jednom funkcijom cilja, dok je u simulacijama iz druge grupe, rešavan višekriterijumski model raspoređivanja sa ograničenjima, koji je detaljno opisan u Poglavlju 4.

Obe grupe simulacija su rađene u CloudSim 3.0.3 okruženju [21]. Detalji o CloudSim platformi dati su u Poglavlju 7.4.2. Takođe, obe simulacije izvršavane su na računarskoj platformi sa Intel CoreTM i7-4770K procesorom @4GHz sa 32GB RAM memorije, Windows 10 Professional x64 operativnim sistemom i Java Development Kit 11 (JDK 11) i IntelliJ integrisanim razvojnim okruženjem.

Potencijalna rešenja problema u obe simulacije kodirana su kao skup zahteva krajnjih korisnika, gde je za svaki zahtev mapirana odgovarajuća virtuelna mašina, kao i u slučaju kodiranja potencijalnih rešenja hibridne MBO-ABC metaheuristike. Dužina rešenja je ukupan broj prihvaćenih zahteva ili poslova u klaud sistemu. Više detalja o kodiranju potencijalnih rešenja može se pročitati u Poglavlju 7.4.1.

U nastavku su prvo prikazane simulacije sa pravim skupom podataka.

8.4.1 Simulacije sa pravim skupom podataka

Korišćenjem realnih podataka generisanih iz robusnih logova super računara je najbolji način da se izvrši evaluacija performansi metaheuristika za rešavanje problema raspoređivanja poslova u klaud računarstvu. U simulacijama sa pravim skupom podataka (realni podaci) rešavan je isti model kao u [34].

Karakteristike realnih zahteva koji su korišćeni u simulacijama generisani su iz NASA Ames iPSC/860 log fajla [76] u Feitelson Parallel Workloads arhivi (PWA). NASA iPSC se nalazi u numeričkom aerodinamičkom simulacionom sistemu, divizije u NASA Ames istraživačkom centru. Centar sadrži podatke iz perioda oktobar 1993. - decembar 1993. za 128-čvorova iPSC/860 klastera super računara. Zapisi

sadrže podatke o 42264 zahteva i 128 resursa (CPU). Podaci su preuzeti iz datoteke evidencije praćenja opterećenja *NASA-iPSC-1993-3.swf* sa sledeće URL adrese: `https://www.cse.huji.ac.il/labs/parallel/workload/l_nasa_ipsc/`. Iz ovih zapisa su preuzeti ID zahteva, dužina svakog zahteva izraženog u MIPS jedinicama i potreban broj procesirajućih elemenata (eng. requested processing elements (PEs)) za obradu svakog zahteva.

Kao što je i navedeno, u simulacijama je generisana klaud infrastruktura u CloudSim simulatoru. U svim simulacijama korišćene su homogene virtuelne mašine, koje se nalaze na dva fizička host računara licranih u jednom data centru. Data centar je generisan sa osnovnim CloudSim karakteristikama. Karakteristike host računara i virtuelnih mašina korišćenih u eksperimentima prikazani su u tabelama 21 [114] i 22 [114], respektivno.

Tabela 21: *CloudSim konfiguracija host računara u simulacijama WOA-AEFS metaheuristike sa pravim skupom podataka*

Identifikator hosta	**Parametar**	**Vrednost**
Host1	RAM	3 (jedinice: GB)
	CPU tip	Intel Core 2 Extreme X6800
	Broj jezgara (PEs)	2
	CPU mogućnosti	27079 (jedinice: MIPS)
	Kapacitet skladišnog prostora	1 (jedinice: TB)
	Mrežni protok	10 (jedinice: Gbps)
	VM polisa raspoređivanja	bazirana na vremenu
Host 2	RAM	3 (jedinice: GB)
	CPU tip	Intel Core i7 Extreme Edition 3960X
	Broj jezgara (PEs)	6
	CPU mogućnost	177730 (jedinica: MIPS)
	Kapacitet skladištenja	1 (jedinica: TB)
	mrežni protok	10 (jedinica: Gbps)
	VM polisa raspoređivanja	bazirana na vremenu

Tabela 22: *CloudSim konfiguracija virtuelnih mašina u simulacijama WOA-AEFS metaheuristike sa pravim skupom podataka*

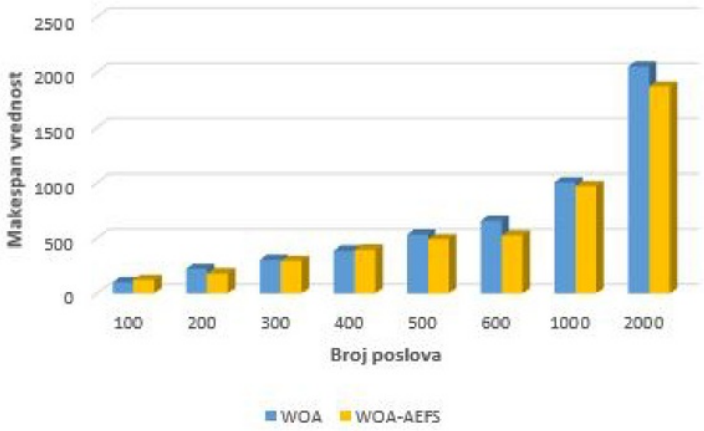
Parametar	Vrednost
broj virtuelnih mašina	10
CPU snaga obrade	9726 (jedinica: MIPS)
RAM memorija	512 (jedinica: MB)
kapacitet mrežnog protoka	1000 (jedinica: Mbps)
kapacitet skladištenja	10 (jedinica: GB)
polisa raspoređivanja poslova	bazirana na vremenu
VMM (Hipervizor)	Xen
Operativni sistem	Linux
broj CPU	1
CPU tip	Pentium 4 Extreme Edition

Sa ciljem validiranja skalabilnosti WOA-AEFS metaheuristike, izvršene su simulacije sa malim skupom zahteva (od 100 do 600 poslova) i sa velikim skupom zahteva (od 1000 do 2000 poslova).

Osnovni WOA-AEFS parametri su podešeni na sledeći način: veličina populacije na 30 ($N = 30$) i maksimalni broj iteracija u jednom puštanju algoritma na 1000 ($maxIter = 1000$). Parametar *limit* je podešen na vrednost 33 ($round(1000/30)$), dok su promenljivi parametri $\vec{a}$, eir i α dinamički podešavani tokom jednog puštanja algoritma korišćenjem izraza 61, 69 i 72, respektivno. Drugi WOA-AEFS parametri su podešeni kao što je prikazano u Tabeli 18.

Isto CloudSim okruženje, kao i parametri veličine populacije i maksimalni broj iteracija su korišćeni u [34]. Na ovaj način, moguće je bilo sprovesti realniju komparativnu analizu sa pristupima koji su prikazani u [34].

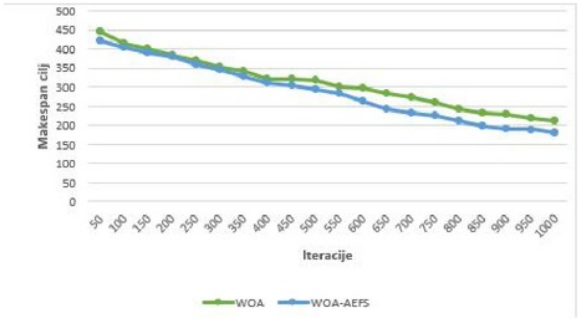
Prvo je urađena komparativna analiza sa osnovnom WOA metaheuristikom. Rezultati su prikazani na Slici 22 [114].



Slika 22: *Komparativna analiza—WOA-AEFS vs. WOA korišćenjem NASA iPSC realnih podataka za mali i veliki broj poslova*

Na osnovu prikazanog dijagrama (Slika 22), jasno je da je WOA-AEFS postiže značajno bolje performanse od originalnog WOA, osim u simulacijama sa 400 zahteva. U ovom slučaju, originalni WOA algoritam postiže neznatno bolje performanse makespan indikatora (oko 1%). Najznačajnija poboljšanja u performansama mogu da se primete u simulacijama sa 600 zahteva, gde je WOA-AEFS bolji od originalne WOA metaheuristike za više od 25 %.

Graf brzine konvergencije WOA-AEFS i WOA metaheuristika za test instance sa 200 poslova je dat na Slici 23 [114].



Slika 23: *Grafikon brzine konvergencije WOA-AEFS vs. WOA korišćenjem NASA iPSC realnih podataka za 200 zahteva*

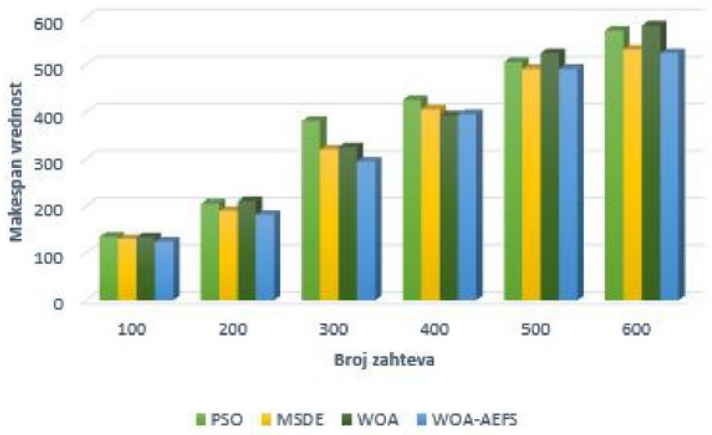
Da bi se još bolje izmerile performanse i potvrdila robusnost WOA-AEFS metaheuristike, izvršena je komparativna analiza sa algoritmima MSDE (moth search differential evolution) i PSO, čiji su rezultati prezentovani u [34]. U ovom radu su

takođe prikazani i rezultati nekih heurističkih metoda (round robin - RR i shortest job first - SJF), ali su one isključene iz analize, pošto postižu značajno lošije performanse od metaheurističkih pristupa.

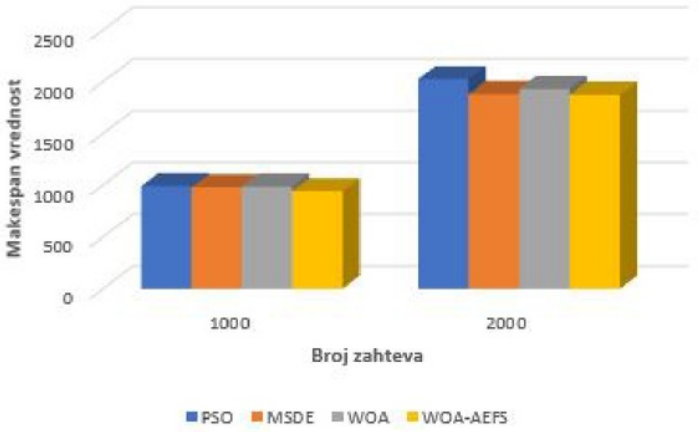
U [34], rezultati originalnog MS algoritma su takođe prikazani. Međutim, i ovaj algoritam je isključen iz komparativne analize, obzirom da MSDE značajno nadmašuje osnovni MS algoritam. Međutim, da bi se ustanovile performanse originalog WOA u odnosu na MSDE i PSO, osnovna WOA verzija je prikazana u komparativnoj analizi. Rezultati osnovnog WOA su takođe prikazani u radu [34].

Treba napomenuti da rezultati originalnog WOA nisu preuzeti iz [34], već je i originalni WOA adaptiran i implementiran za potrebe istraživanja prikazanog u ovoj disertaciji.

Rezultati komparativne analize sa manjim brojem zahteva (od 100 do 600) i sa većim brojem poslova prikazani su na slikama 24 [114] i 25 [114], respektivno.



Slika 24: *Komparativna analiza—WOA-AEFS vs. drugi algoritmi korišćenjem NASA iPSC realnih podataka sa manjim brojem poslova*



Slika 25: *Komparativna analiza—WOA-AEFS vs. drugi algoritmi korišćenjem NASA iPSC realnih skupova sa većim brojem poslova*

Na osnovu rezultata koji se mogu videti na dijagramima sa slika 24 i 25, može da se zaključi da u proseku, WOA-AEFS ostvaruje bolje makespan vrednosti nego drugi pristupi uključeni u komparativnu analizu. Sa dijagrama se takođe može primetiti da WOA-AEFS metaheuristika generiše rešenja istog kvaliteta kao MSDE u simulacijama sa 500 poslova. U simulacijama sa većim brojem poslova (1000 i 2000), u oba slučaja WOA-AEFS nadmašuje sve druge pristupe.

Poboljšanja makespan funkcije cilja, koju WOA-AEFS ostvaruje u odnosu na druge pristupe, iskazane u procentima date su u Tabeli 23 [114].

Tabela 23: *Poboljšanja WOA-AEFS algoritma u odnosu na druge algoritme u simulacijama sa NASA iPSC realnim podacima (vrednosti su iskazane u procentima)*

Broj poslova	PSO	MSDE	WOA
100	+8.60%	+4.83%	+7.63%
200	+13.23%	+4.97%	+16.16%
300	+29.08%	+8.47%	+10.18%
400	+7.71%	+2.53%	−0.70%
500	+2.99%	+0.02%	+6.96%
600	+9.19%	+1.52%	+11.08%
1000	+1.89%	0.92%	1.58%
2000	+8.10%	+0.42%	+3.08%

U poslednjem setu eksperimenata sa realnim skupom podataka, da bi se utvrdile performanse WOA-AEFS metaheuristike u balansiranju opterećenja, prikazana je komparativna analiza između WOA-AEFS i originalnog WOA, MSDE i PSO za vrednost DI indikatora. Za više informacija, o DI indikatori, pogledati Poglavlje 4.2.

Komparativna analiza sa DI indikatorom je data u Tabeli 24 [114], gde su rezultati za MSDE i PSO preuzeti iz [34]. Najbolji rezultati iz svake kategorije instanci testa su markirani u bold stilu.

Tabela 24: *Komparativna analiza DI indikatora u simulacijama sa NASA iPSC realnim skupom podataka*

Broj poslova	PSO	MSDE	WOA	WOA-AEFS
100	0.974118	0.972773	1.004538	**0.935002**
200	1.375412	1.138444	1.337259	**1.373259**
300	1.051463	**0.905947**	1.032713	0.983205
400	1.050034	**0.878397**	1.063955	0.883302
500	1.077579	0.942521	1.311571	**0.941056**
600	1.097044	0.983239	1.270371	**0.976200**
1000	**1.013558**	1.040253	1.093521	1.0387226
2000	**0.974118**	1.011911	0.991087	0.9835211
Bolji	2	2	0	3

Na osnovu analiza rezultata DI indikatora, prikazan u Tabeli 24, može da se primeti da je u proseku, WOA-AEFS ostvario najbolje rezultate. PSO algoritam je pokazao da je sposoban za uspostavljanje efikasnog balansiranja opterećenja između virtuelnih mašina u okruženju sa velikim brojem poslova (1000 i 2000), i u ovim testovima PSO je pokazao najbolje performanse. U simulacijama sa 300 i 400 poslova, MSDE je nadmašio sve algoritme, uključujući i WOA-AEFS.

Predloženi WOA-AEFS je pokazao superiorniji kvalitet rešenja u simulacijama sa malim test instancama, i to sa 100,200,500 i 600 zahteva krajnjih korisnika, u simulacijama za balansiranje opterećenja.

Kao opšti zaključak navodi se da u proseku, WOA-AEFS ostvaruje najbolje vrednosti DI indikatora nego sve druge metaheuristike uključene u komparativnu analizu. Drugi po redu je MSDE pristup, dok su originalni WOA i PSO u proseku generisali slične rezultate.

Razmatrajući mogućnost implementiranja tehnika metaheuristika za problem raspoređivanja poslova u realnom klaud okruženju, vreme izvršavanja algoritama predstavlja značajan pokazatelj (CPU vreme). Obzirom da u [34], detalji platforme izvršavanja nisu dati, ne može da se poredi potrebno CPU vreme WOA-AEFS sa CPU vremenom MSDE i PSO. Međutim, obzirom da su WOA i WOA-AEFS implementirani i testirani na istoj računarskoj platformi, u ovom slučaju može da se napravi komparativna analiza vremena izvršavanja ova dva algoritma. Komparacija CPU vremena je prezentovana u Tabeli 25 [114].

Tabela 25: *Komparativna analiza CPU vremena u simulacijama sa NASA iPSC realnim skupom podataka*

Algoritam	Broj poslova							
	100	200	300	400	500	600	1000	2000
WOA	**2730**	4122	6541	**8567**	11005	**15349**	**35152**	118782
WOA-AEFS	2855	**3983**	**6374**	8750	**10730**	15765	36110	**109351**

Na osnovu prikazanih rezultata može da se zaključi je vreme izvršavanja obe metaheuristike približno jednako. Ovo ima važne implikacije za problem raspoređivanja poslova u klaud okruženju, kao i na performanse metaheuristika, s obzirom na to da kompleksnost WOA-AEFS u odnosu na originalni WOA nema značajan uticaj na vreme izvršavanja.

8.4.2 Simulacije sa veštačkim skupom podataka

U drugom setu eksperimenata raspoređivanja poslova na klaudu, korišćen je višekriterijumski model sa performansama i budžetskim ograničenjima. U ovom modelu, uzeta je u obzir minimizacija makespan indikatora, kao i minimizacija ukupnih troškova raspoloživog budžeta. Sličan model korišćen je i u simulacijama, čiji su rezultati prikazani u radovima [102] i [92], koji su objavljeni u vrhunskim međunarodnim časopisima.

Ovaj model je u literaturi iz domena računarstva poznat kao model raspoređivanja resursa baziran na performansama i ograničenjima raspoloživog budžeta. Više informacija o ovom modelu može se naći u Poglavlju 4.3.

U ovom skupu simulacija korišćeni su veštački podaci, generisani u okviru CloudSim 3.0.3 platforme. Za svako puštanje algoritma generisano je nasumično okruženje sa 50 heterogenih virtuelnih mašina, koje su implementirane „iznad" host računara u data centrima. Uzet je u obzir različit broj heterogenih poslova (od 100 do 500) različitih dužina. Osim ovoga, u simulacijama je razmatrana i stopa pristizanja različitih zahteva, tačnije razmatrane su performanse sistema kada pristiže 10 i 40 zahteva u jedinici vremena. Isto simulaciono okruženje je korišćeno u [92].

Karakteristike poslova, virtuelnih mašina i data centara su prikazani u Tabeli 26 [114] i u Tabeli 27 [114].

Tabela 26: *Karakteristike virtuelnih mašina, host računara i data centara u eksperimentima sa veštačkim skupom podataka (WOA-AEFS algoritam)*

Naziv entiteta	Parametar	Vrednost
VM	Broj virtuelnih mašina	50
	RAM	512 (jedinica: MB)
	CPU snaga obrade	1860,2660 (jedinica: MIPS)
	Kapacitet skladištenja	1 (jedinica: GB)
	Kapacitet mrežnog protoka	1000 (jedinica: Mbps)
	Polisa raspoređivanja poslova	bazirana na vremenu
	VMM (Hipervizor)	Xen
	Operativni sistem	Linux
	Broj CPU	1
Host	RAM	2 (jedinica: GB)
	Kapacitet skladištenja	10 (jedinica: GB)
	Kapacitet mrežnog protoka	1 (jedinica:Gbps)
	Polisa raspoređivanja virtuelnih mašina	bazirana na prostoru
Data centar	Broj data centara	10
	Broj host računara	10

Tabela 27: *Karakteristike poslova u eksperimentima sa veštačkim skupom podataka (WOA-AEFS algoritam)*

Parametar	Vrednost
broj poslova	100–500
dužina posla	400–1000 (jedinica: MI)
veličina fajla	200-1000 (jedinica: MB)
veličina izlaza (memorije)	20–40 (jedinica: MB)

Osnovni kontrolni parametri WOA-AEFS metaheuristike podešeni su na sledeći način: veličina populacije na 30 ($N = 30$) i maksimalni broj iteracija u jednom puštanju algoritma na 1000 ($maxIter = 1000$). Vrednost parametra $limit$ je podešena na 33 ($round(1000/30)$), dok su vrednosti dinamičkih parametara $\vec{a}$, eir i α adaptirani u toku jednog puštanja algoritma korišćenjem jednačina 61, 69 i 72, respektivno. Ostali WOA-AEFS parametri su podešeni kao što je prikazano u Tabeli 18.

Slično kao i u radu [92], odvojeno je analizirana makespan funkcija cilja, trošak i narušavanje roka izvršenja svih zadataka (eng. deadline violation constraint) i prikazana je komparativna analiza sa Min-Min i FCFS (first come first serve) heuristikama, kao i metaheuristikama CSPSO (cuckoo search particle swarm optimization) i PBACO (performance budget ACO).

Slično kao i u prethodnim simulacijama, da bi se bolje evaluirale performanse WOA-AEFS, u komparativnu analizu je uključen i osnovni WOA algoritam. Rezultati

simulacija Min-Min i FCFS heuristike, kao i CPSO i PBACO metaheuristike su preuzete iz [92]. Osnovni WOA pristup je implementiran za potrebe prikazanog istraživanja u Java tehnologiji i izvršeni su odgovarajući eksperimenti. U svakoj simulaciji, WOA i WOA-AEFS su evaulirani u 100 nezavisnih puštanja i prosečni rezultati su zabeleženi. Osim toga, za svako puštanje algoritama, inicijalizovano je novo slučajno klaud okruženje sa karakteristikama prikazanim u tabelama 26 i 27.

Zbog bolje preglednosti, rezultati simulacija kategorisani su u tri grupe: makespan funkcija cilja, troškovi izvršavanja poslova i narušavanje roka izvršavanja zadataka.

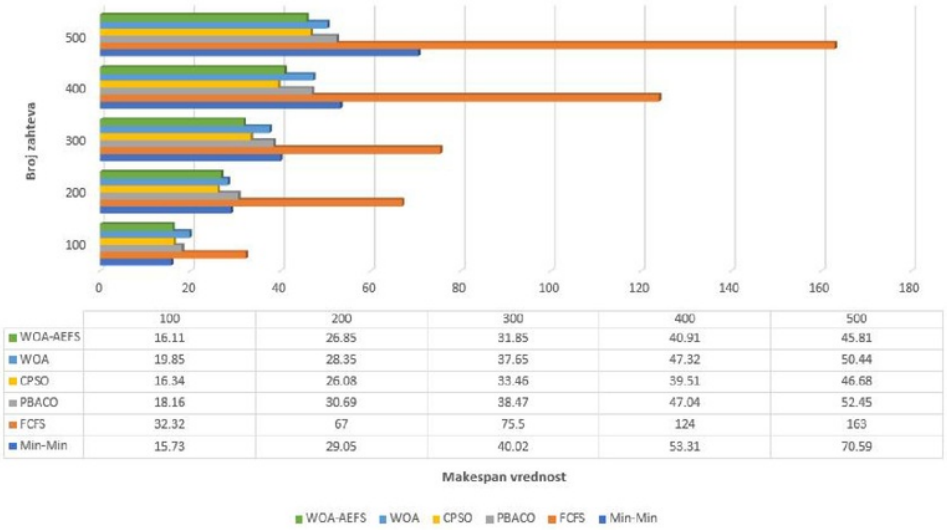
Analiza makespan funkcije cilja

Rezultati komparativne analize (dijagram i podaci) između WOA-AEFS i drugih pristupa za vrednost makespan funkcije cilja, kada je stopa pristizanja poslova podešena na 10 prikazana je na Slici 26 [114].

Na osnovu prikazanih rezultata, zaključuje se da je u proseku WOA-AEFS algoritam ostvario najbolju vrednost makespan indikatora u odnosu na sve druge pristupe uključene u analizu. Kao druga najbolja metaheuristika pokazao se CPSO algoritam, koja je u isto vreme postiže mnogo bolje performanse od osnovnog WOA. Metaheuristika WOA-AEFS je ostvarila bolje rezultate nego CPSO algoritam u test instancama sa 100, 300 i 500 poslova. S druge strane, u simulacijama sa 200 i 400 zahteva, CPSO je ostvario bolje makespan vrednosti nego predloženi WOA-AEFS.

Takođe, na osnovu prikazanih rezultata, može da se izvede zaključak da je WOA-AEFS značajno nadmašio osnovni WOA algoritam u svim test slučajevima. Najveći rast performansi može da se primeti kod problema sa 100 poslova, gde je WOA-AEFS bolji od WOA za oko 19 %. U testovima sa 200, 300, 400 i 500 poslova, WOA-AEFS je poboljšao vrednost makespan indikatora originalnog WOA za oko 5%, 15%, 13% i 9%, respektivno.

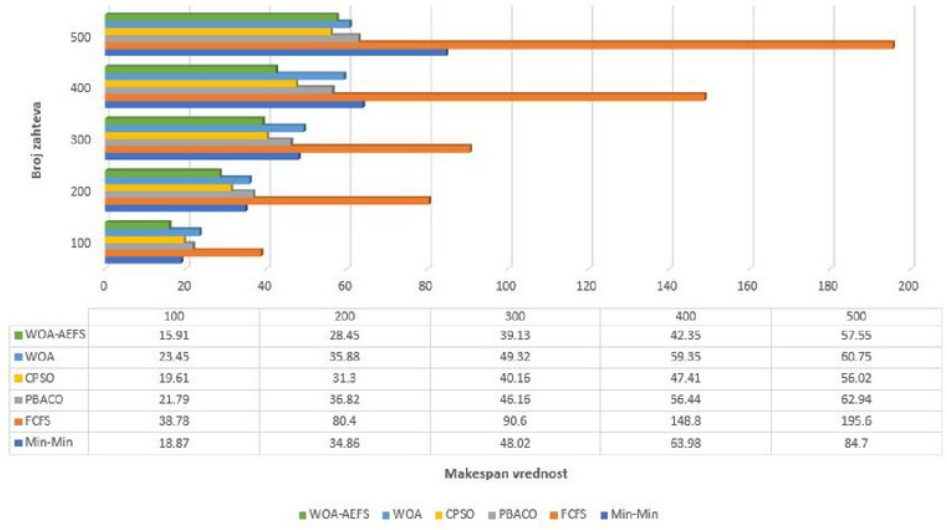
Na osnovu rezultata takođe se vidi da osnovni WOA generiše rešenja sličnog kvaliteta kao PBACO metaheuristika.



	100	200	300	400	500
WOA-AEFS	16.11	26.85	31.85	40.91	45.81
WOA	19.85	28.35	37.65	47.32	50.44
CPSO	16.34	26.08	33.46	39.51	46.68
PBACO	18.16	30.69	38.47	47.04	52.45
FCFS	32.32	67	75.5	124	163
Min-Min	15.73	29.05	40.02	53.31	70.59

Slika 26: *Komparativna analiza makespan funkcije cilja — WOA-AEFS vs. druge tehnike korišćenjem veštačkog skupa podataka (stopa dolaska 10)*

Rezultati komparativne analize (dijagram i podaci), kada je stopa pristizanja poslova podešena na 40 je prikazana na Slici 27 [114].



	100	200	300	400	500
WOA-AEFS	15.91	28.45	39.13	42.35	57.55
WOA	23.45	35.88	49.32	59.35	60.75
CPSO	19.61	31.3	40.16	47.41	56.02
PBACO	21.79	36.82	46.16	56.44	62.94
FCFS	38.78	80.4	90.6	148.8	195.6
Min-Min	18.87	34.86	48.02	63.98	84.7

Slika 27: *Komparativna analiza makespan funkcije cilja — WOA-AEFS vs. druge algoritme korišćenjem veštačkog skupa podataka (stopa pristizanja 40)*

U eksperimentima sa stopom pristizanja zahteva od 40 klaudleta u jedinici vremena, WOA-AEFS je ostvario bolje rezultate nego CPSO u 4 od 5 test instanci. Samo u testovima sa 500 zahteva, CPSO je nadmašio WOA-AEFS za oko 2%.

Najznačajnije razlike u performansama između ova dva algoritma mogu da se uoče u testu sa 100 zahteva, gde je WOA-AEFS ostvario bolje rezultate za 19%.

U simulacijama gde je stopa pristizanja poslova podešena na 40, WOA-AEFS je dokazao da je najbolja metaheuristika, dok je CPSO na drugom mestu. U svim test instancama, WOA-AEFS je nadmašio originalnu WOA metaheuristiku, koja je generisala slične rezultate kao PBACO.

Kao što je i očekivano, u simulacijama sa obe stope dolaska zahteva (10 i 40), heuristički algoritmi su pokazali najgore performanse. Međutim, za pojedine test instance, Min-Min heuristika je ostvarila obećavajuće rezultate. Tako na primer, u testovima sa 100 i 200 klaudleta i obe stope pristizanja poslova (10 i 40), Min-Min heuristika je ostvarila bolje performanse nego PBACO algoritam. S druge strane, kada se uzmu u obzir svi izvršeni testovi, Min-Min heuristika je u proseku ostvarila dva puta bolje performanse nego FCFS heuristika.

Analiza troškova

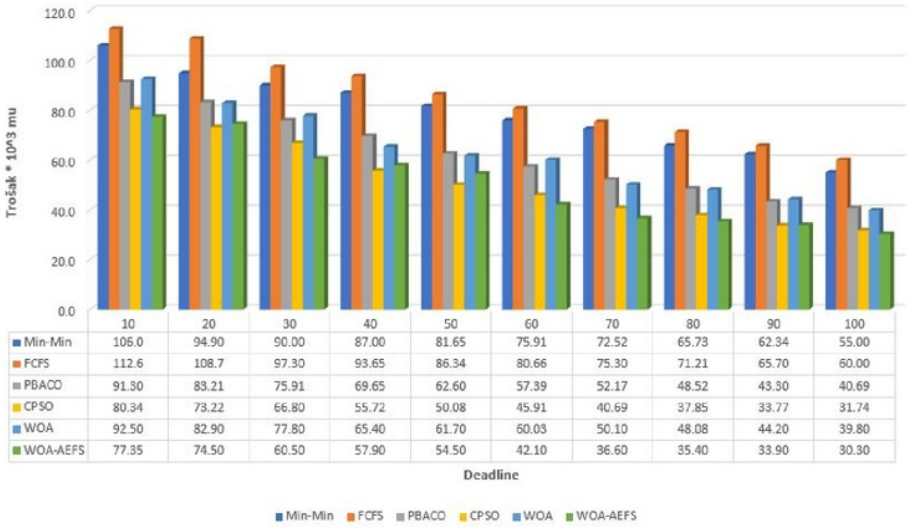
Kao u [92], u simulacijama je posebno razmatrana minimizacija funkcije cilja ukupnih troškova, gde su sprovedeni eksperimenti sa 200, 400 i 500 zahteva (zadataka) krajnjih korisnika i promenljivim rokovima izvršavanja (od 10 do 100). Rezultati komparativne analize sa drugim pristupima za 200, 400 i 500 zadataka prikazani su na Slici 28 [114], Slici 29 [114] i Slici 30 [114], respektivno.

Na svim prikazanim grafikonima, vertikalna osa prikazuje ukupan trošak izražen u monetarnim jedinicama, dok horizontalna osa označava rokove izvršavanja zadataka.

Kada se analiziraju rezultati predstavljeni u Slikama 28, 29 i 30, slično kao u analizi makespan funkcije cilja, može da se zaključi da u proseku, predloženi WOA-AEFS ostvaruje najbolje performanse. Drugi najbolji rezultat postiže metaheuristika CPSO [92], koja je u samo nekoliko test instanci nadmašila predloženi WOA-AEFS. Neke od ovih instanci uključuju: 200 zahteva sa rokovima od 20, 40, 50 i 90, 400 zahteva sa rokovima od 40,50,70 i 90, kao i 500 zahteva sa rokovima od 10, 70 i 100.

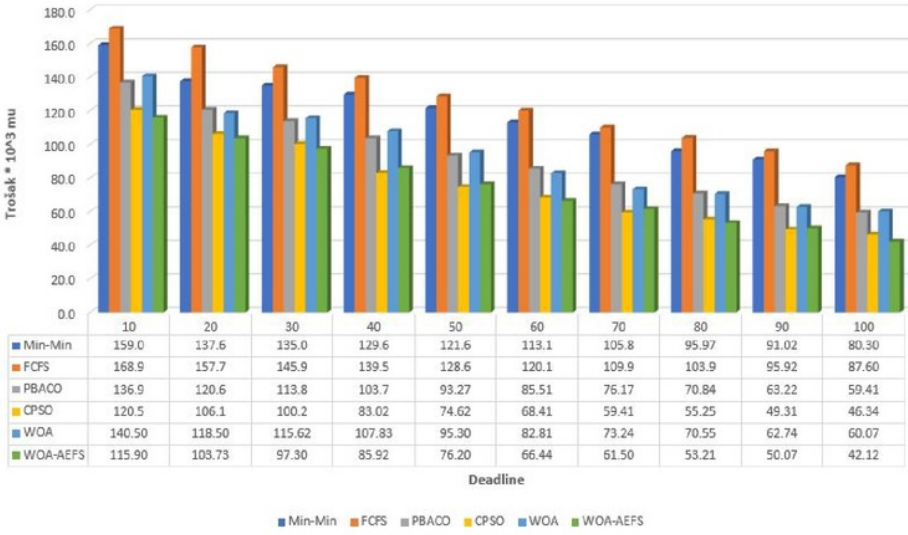
Slično kao u prethodnim analizama, predloženi WOA-AEFS je za sve test instance ostvario mnogo bolje rezultate nego originalni WOA, i u isto vreme postigao bolji kvalitet rešenja i bolju konvergenciju. Osnovni WOA algoritam je ostvario slične performanse kao PBACO metaheuristika.

Kada se analiziraju ukupni troškovi, oba heuristička pristupa su postigla značajno lošije rezultate od metaheurističkih algoritama.



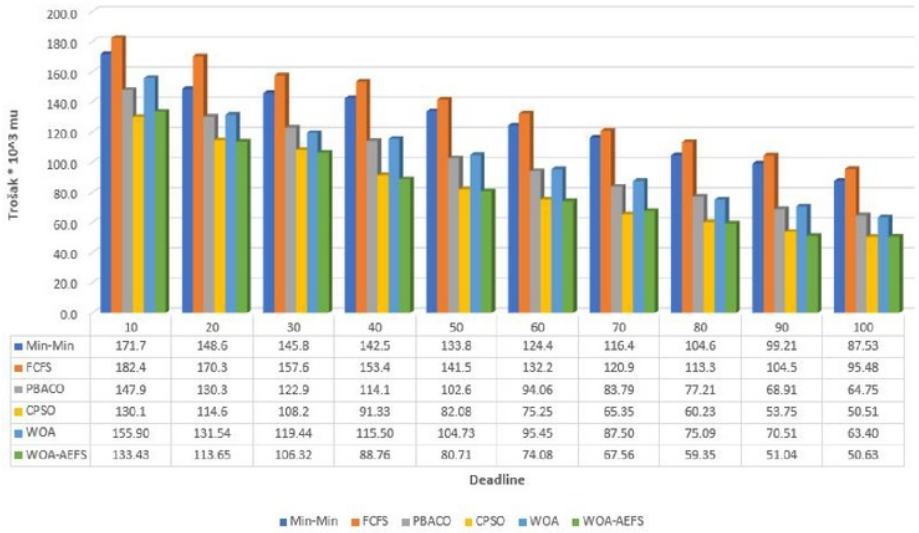
	10	20	30	40	50	60	70	80	90	100
Min-Min	106.0	94.90	90.00	87.00	81.65	75.91	72.52	65.73	62.34	55.00
FCFS	112.6	108.7	97.30	93.65	86.34	80.66	75.30	71.21	65.70	60.00
PBACO	91.30	83.21	75.91	69.65	62.60	57.39	52.17	48.52	43.30	40.69
CPSO	80.34	73.22	66.80	55.72	50.08	45.91	40.69	37.85	33.77	31.74
WOA	92.50	82.90	77.80	65.40	61.70	60.03	50.10	48.08	44.20	39.80
WOA-AEFS	77.35	74.50	60.50	57.90	54.50	42.10	36.60	35.40	33.90	30.30

Slika 28: *Komparativna analiza troškova sa 200 zahteva i promenljivim rokom izvršavanja zadataka —WOA-AEFS vs. drugi algoritmi korišćenjem veštačkog skupa podataka*



	10	20	30	40	50	60	70	80	90	100
Min-Min	159.0	137.6	135.0	129.6	121.6	113.1	105.8	95.97	91.02	80.30
FCFS	168.9	157.7	145.9	139.5	128.6	120.1	109.9	103.9	95.92	87.60
PBACO	136.9	120.6	113.8	103.7	93.27	85.51	76.17	70.84	63.22	59.41
CPSO	120.5	106.1	100.2	83.02	74.62	68.41	59.41	55.25	49.31	46.34
WOA	140.50	118.50	115.62	107.83	95.30	82.81	73.24	70.55	62.74	60.07
WOA-AEFS	115.90	103.73	97.30	85.92	76.20	66.44	61.50	53.21	50.07	42.12

Slika 29: *Komparativna analiza troškova sa 400 zahteva promenljivim rokom izvršavanja zadataka—WOA-AEFS vs. drugi algoritmi korišćenjem veštačkog skupa podataka*



	10	20	30	40	50	60	70	80	90	100
Min-Min	171.7	148.6	145.8	142.5	133.8	124.4	116.4	104.6	99.21	87.53
FCFS	182.4	170.3	157.6	153.4	141.5	132.2	120.9	113.3	104.5	95.48
PBACO	147.9	130.3	122.9	114.1	102.6	94.06	83.79	77.21	68.91	64.75
CPSO	130.1	114.6	108.2	91.33	82.08	75.25	65.35	60.23	53.75	50.51
WOA	155.90	131.54	119.44	115.50	104.73	95.45	87.50	75.09	70.51	63.40
WOA-AEFS	133.43	113.65	106.32	88.76	80.71	74.08	67.56	59.35	51.04	50.63

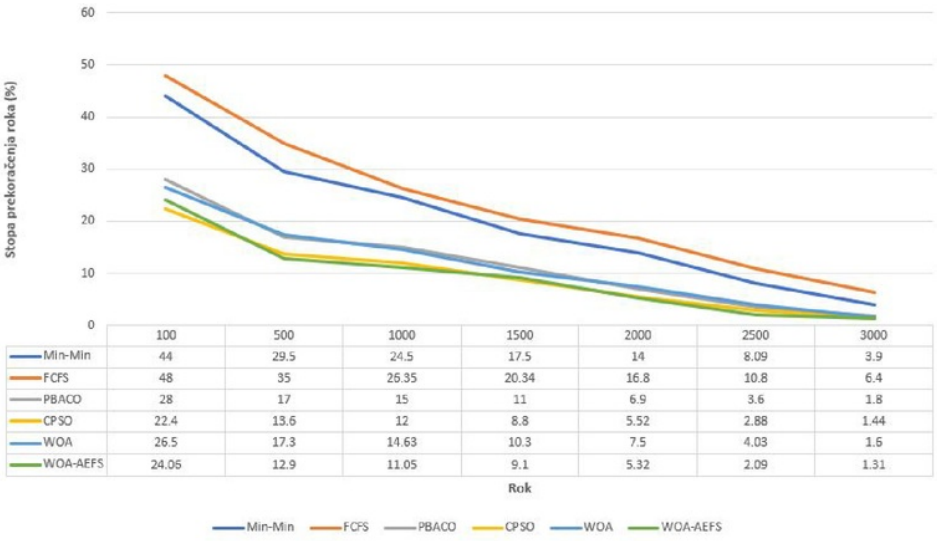
Slika 30: *Komparativna analiza troškova sa 500 zahteva i promenljivim rokom izvršavanja zadataka —WOA-AEFS vs. drugi algoritmi korišćenjem veštačkog skupa podataka*

Analiza narušavanja roka za izvršavanje zadataka

Poslednja analiza koja je urađena u simulacijama sa veštačkim skupom podataka je evaluiranje stope narušavanja roka. Glavni cilj ove analize je ocena kvaliteta usluga (QoS) predložene metaheuristike.

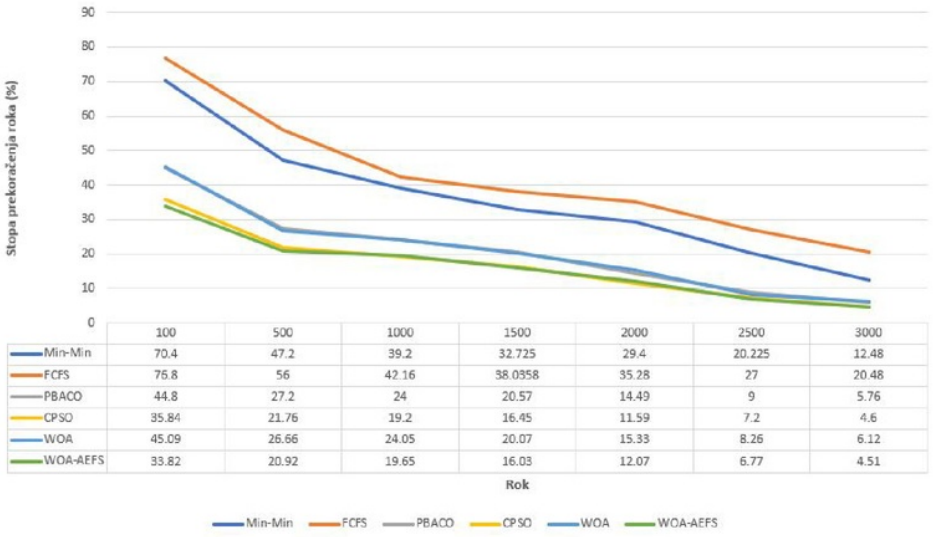
Simulacije su sprovedene sa 200, 400 i 500 zahteva i promenljivim rokovima izvršavanja zadataka (od 100 do 3000), kao i u radu [92]. Rezultati komparativne analize narušavanja roka sa 200, 400 i 500 zahteva sa ograničenjima roka izvršavanja zadataka su prikazani na Slici 31 [114], Slici 32 [114] i Slici 33 [114], respektivno.

U svim prezentovanim grafikonima stopa narušavanja prikazana je u procentima na ordinatnoj osi, dok su rokovi izvršavanja zadataka prikazani na apscisi.



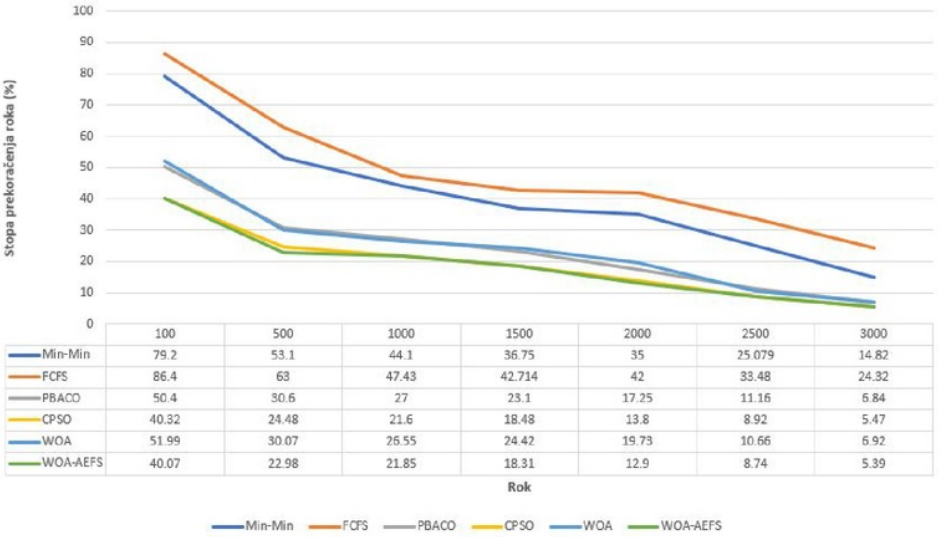
	100	500	1000	1500	2000	2500	3000
Min-Min	44	29.5	24.5	17.5	14	8.09	3.9
FCFS	48	35	26.35	20.34	16.8	10.8	6.4
PBACO	28	17	15	11	6.9	3.6	1.8
CPSO	22.4	13.6	12	8.8	5.52	2.88	1.44
WOA	26.5	17.3	14.63	10.3	7.5	4.03	1.6
WOA-AEFS	24.06	12.9	11.05	9.1	5.32	2.09	1.31

Slika 31: *Komparativna analiza narušavanja roka sa 200 zahteva —WOA-AEFS vs. drugi algoritmi korišćenjem veštačkog skupa podataka*



	100	500	1000	1500	2000	2500	3000
Min-Min	70.4	47.2	39.2	32.725	29.4	20.225	12.48
FCFS	76.8	56	42.16	38.0358	35.28	27	20.48
PBACO	44.8	27.2	24	20.57	14.49	9	5.76
CPSO	35.84	21.76	19.2	16.45	11.59	7.2	4.6
WOA	45.09	26.66	24.05	20.07	15.33	8.26	6.12
WOA-AEFS	33.82	20.92	19.65	16.03	12.07	6.77	4.51

Slika 32: *Komparativna analiza narušavanja roka sa 400 zahteva —WOA-AEFS vs. drugi algoritmi korišćenjem veštačkog skupa podataka*



	100	500	1000	1500	2000	2500	3000
Min-Min	79.2	53.1	44.1	36.75	35	25.079	14.82
FCFS	86.4	63	47.43	42.714	42	33.48	24.32
PBACO	50.4	30.6	27	23.1	17.25	11.16	6.84
CPSO	40.32	24.48	21.6	18.48	13.8	8.92	5.47
WOA	51.99	30.07	26.55	24.42	19.73	10.66	6.92
WOA-AEFS	40.07	22.98	21.85	18.31	12.9	8.74	5.39

Slika 33: *Komparativna analiza narušavanja roka sa 500 zahteva —WOA-AEFS vs. drugi algoritmi korišćenjem veštačkog skupa podataka*

Na osnovu rezultata prikazanih na slikama 31, 32 i 33 vidi se da je u proseku WOA-AEFS superirniji od svih drugih heuristika i metaheuristika koje su obuhvaćene komparacijom.

Slično kao i u prethodna dva testa, druga metaheuristika koja najmanje narušava rokove za izvršavanje zadataka je CPSO. Takođe, kao i u prethodnim testiranjima, osnovni WOA pristup je ostvario slične rezultate kao i PBACO metaheuristika. Oba heuristička pristupa su ostvarila mnogo lošije rezultate od svih metaheuristika.

Kao sveobuhvatni zaključak, kada se uzmu u obzir sve tri analize - makespan funkcije cilja, minimizacije ukupnih troškova i vrednost stope narušavanje rokova, u simulacijama sa veštačkim skupom podataka, navodi se da u proseku WOA-AEFS postiže bolje performanse od svih drugih algoritama. Drugi najbolji pristup je CSPSO, dok osnovni WOA postiže slične performanse kao PBACO algoritam. Heuristika FCFS postiže najgore rezultate, dok u pojedinim testovima performanse Min-Min heuristike mogu da se mere sa performansama metaheurističkih metoda.

8.5 WOA-AEFS zaključak

U ovom delu disertacije detaljno je prikazana implementacija hibridnog WOA-AEFS algoritma, koji prevazilazi uočene nedostatke originalnog WOA pristupa.

Algoritam je prvo testiran na skupu klasičnih benčmark funkcija, koji se sastoji iz 23 problema globalne optimizacije bez ograničenja, a zatim su izvršeni testovi na modelu raspoređivanja poslova u klaud okruženju sa jednom funkcijom i više funkcija cilja, i to sa veštačkim i pravim skupovima podataka.

Za sve instance testova, urađena je komparativna analiza sa osnovnim MBO algoritmom i sa drugim vrhunskim pristupima, koji su testirani na istim instancama problema i čiji su rezultati publikovani u vrhunskim međunarodnim časopisima.

Na osnovu rezultata istraživanja, zaključeno je da sledeće:

1. hibridizovani WOA postiže bolje performanse od osnovne WOA verzije, čime je osnovna WOA implementacija unapređena i

2. hibridni WOA-AEFS algoritam generiše bolja rešenja problema raspoređivanja poslova i balansiranja opterećenja od drugih heurisitka i metaheuristika, čime je Hipoteza X dokazana (pogledati Glavu 1).

Rezultati do kojih se došlo ovim istraživanjem publikovani su u radu u jednom vrhunskom međunarodnom časopisu kategorije M22 [114].

9 Zaključak

U ovoj doktorskoj disertaciji pokazano je da se rešavanje problema raspoređivanja poslova i balansiranja opterećenja u klaud okruženju može dalje unaprediti primenom metaheuristika inteligencije rojeva. Problem raspoređivanja odnosi se na definisanje rasporeda izvršavanja zadataka na ograničenom skupu raspoloživih resursa uzimajući pritom u obzir potencijalna ograničenja i funkciju cilja koju je potrebno optimizovati. Glavni cilj raspoređivanja poslova u klaud okruženju jeste kreiranje rasporeda, na osnovu koga se definiše kada i koji virtuelni resursi će obraditi zahteve krajnjih korisnika.

Problemi raspoređivanja poslova i balansiranja opterećenja u klaud okruženju spadaju u grupu NP teških kombinatornih i/ili globlanih problema sa ili bez ograničenja. Na osnovu publikovanih rezultata u relevantnim literaturnim izvorima, vidi se da su metaheuristike inteligencije rojeva koje spadaju u grupu prirodom-inspirisanih algoritama, uspešno primenjivane, kako na benčmark, tako i na praktične NP teške optimizacione probleme (globalne i kombinatorne) i da mogu da postignu bolje rezultate u smislu brzine konvergencije i kvaliteta rešenja, od drugih metoda, tehnika i algoritama. Isto tako, vidi se da problem raspoređivanja poslova i balansiranja opterećenja na klaudu nije dovoljno rešavan primenom algoritama inteligencije rojeva, što je bio i osnovni motivator za sprovođenje istraživanja koje je prikazano u ovoj disertaciji.

Naučni doprinos ove doktorske disertacije verifikovan je publikovanjem u jednom vrhunskom i jednom međunarodnom časopisu od nacionalnog značaja referisanog u Scopus bazi, poglavlju Springer knjige iz serijala LNCS (Lecture Notes in Computer Science) i u zborniku jednog od svetskih vodećih kongresa iz domena evolutivnog računarstva (Congress on Evolutionary Computation '19 - CEC 2019).

U disertaciji su detaljno prikazani rezultati koji su publikovani u međunarodnim časopisima. Dve metaheuristike rojeva u osnovnoj i unapređenoj (hibridizovanoj) verziji su implementirane i adaptirane za rešavanje dva modela problema raspoređivanja poslova i balansiranja opterećenja na klaudu. Prvi model spada u grupu optimizacionih problema sa jednom funkcijom cilja, gde je uzeta u obzir minimizacija ukupnog vremena izvršavanja svih poslova na raspoloživim klaud računarskim

resursima. Drugi rešavani model pripada kategoriji višekriterijumske optimizacije sa ograničenjima, gde su uzete u obzir dve funkcije cilja: ukupno vreme izvršavanja svih poslova i minimizacija troškova u okviru raspoloživog budžeta.

Osnovne verzije metaheuristika MBO i WOA su prvo unapređene hibridizacijom sa drugim algoritmima. U prvom slučaju, MBO je hibridizovan sa ABC algoritmom i pokazano je da hibridni MBO-ABC algoritam eliminiše uočene nedostatke osnovne MBO metaheuristike. U drugom slučaju, sa ciljem eliminisanja nedostataka, koji se ispoljavaju u vidu preuranjene konvergencije i neusklađenog balansa između eksploatacije eksploracije, osnovnog WOA pristupa, hibridizovani WOA-AEFS inkorporira ABC mehanizam eksploracije i jednačinu pretrage FA algoritma u osnovnu WOA metaheuristiku.

Sledeći svetsku praksu, kada se kreira novi, ili unapredi postojeći algoritam, prvo je potrebno izvršiti testiranja na širem skupu standardnih benčmark problema, oba unapređena algoritma su prvo testirana na standardnim funkcijama globalne optimizacije sa ograničenjima vrednosti parametara. Na osnovu rezultata testiranja, zaključeno je da oba algoritma značajno unapređuju osnovne verzije, ali i da postižu bolje rezultate od drugih vrhunskih metaheuristika čiji su rezultati objavljeni u literaturi iz domena računarskih nauka.

Oba unapređena pristupa (MBO-ABC i WOA-AEFS) testirana su za rešavanje problema raspoređivanja poslova i balansiranja opterećenja na klaudu i to sa veštačkim podacima, generisanim u okviru CloudSim platforme, i sa pravim podacima, koji su preuzeti iz globalno dostupnih benčmark baza. Podešavanja kontrolnih parametara, postavke simulacionog okruženja i ostvareni rezultati detaljno su prikazani. Rezultati hibridizovanih algoritama prvo su upoređeni sa kvalitetom rešenja koja postižu osnovni algoritmi, gde je zaključeno da poboljšani algoritmi postižu značajno bolje rezultate u smislu kvaliteta rešenja i brzine konvergencije.

Takođe je izvršena i detaljna komparativna analiza predloženih hibridnih metaheuristika sa drugim vrhunskim heurističkim i metaheurističkim algoritmima čije su performanse validirane na istim modelima raspoređivanja poslova i balansiranja opterećenja na klaudu, sa istim podacima i pod istim eksperimentalnim uslovima, i čiji su rezultati objavljeni u vrhunskim međunarodnim časopisima. U slučaju obe metaheuristike, dokazano je da predloženi algoritmi unapređuju rezultate do kojih

su drugi algoritmi došli u rešavanju problema raspoređivanja poslova i balansiranja opterećenja na klaudu, čime je osnovna hipoteza istraživanja dokazana.

Na osnovu svega navedenog, u ovoj doktorskoj disertaciji je ostvareno:

- pregled modela raspoređivanja poslova i balansiranja opterećenja u klaud okruženju;
- pregled postojećih metaheurističkih i heurističkih metoda za rešavanje problema raspoređivanja poslova i balansiranja opterećenja u klaud okruženju;
- unapređenje rešavanja problema raspoređivanja poslova i balansiranja opterećenja na klaudu primenom metaheuristika inteligencije rojeva i
- detaljna komparativna analiza navedenih metoda, sa isticanjem prednosti i nedostataka jednih metoda u odnosu na druge.

Glavni naučni doprinos istraživanja koje je prikazano u doktorskoj disertaciji je:

- unapređenje rešavanja problema raspoređivanja poslova i balansiranja opterećenja u klaud okruženju primenom i adaptacijom algoritama inteligencije rojeva, kako osnovnih, tako i unapređenih verzija;
- unapređenje postojećih verzija i implementacija metaheuristika rojeva;
- kritički osvrt na rezultate do kojih dolaze druge metode i tehnike za rešavanje problema raspoređivanja poslova i balansiranja opterećenja na klaudu;
- komparativna analiza rezultata raznih algoritama inteligencije rojeva za ovu klasu problema, kao i između algoritama inteligencije rojeva i drugih metoda i algoritama i
- detaljan prikaz matematičkih modela i formulisanja problema raspoređivanja poslova i balansiranja opterećenja u klaud okruženju.

Prikazani rezultati istraživanja potvrđuju da su dokazane opšta, posebna i pojedinačne hipoteze, koje su predstavljale polaznu osnovu za sprovedeno istraživanje i simulacije.

Budući da je klaud računarstvo dinamična i živa oblast koja je tek počela da se razvija, i s pogledom na činjenicu da sa rastom klaud infrastrukture raste i potreba za rešavanjem problema i izazova koji su razmatrani u ovoj doktorskoj disertaciji, postoji

veliki potencijal za nastavak istraživanja u ovom domenu. U skladu s ovim, kao segment budućeg istraživanja biće adaptirane i implementirane i druge metaheuristike inteligencije rojeva, kako za rešavanje problema raspoređivanja poslova, tako i za druge izazove klaud računarstva.

10 Literatura

[1] Cloud computing – statistics on the use by enterprises. https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Cloud_computing_-_statistics_on_the_use_by_enterprises, accessed: 2019-10-24

[2] Gartner forecasts worldwide public cloud revenue to grow 17.5 percent in 2019. https://www.gartner.com/en/newsroom/press-releases/2019-04-02-gartner-forecasts-worldwide-public-cloud-revenue-to-g, accessed: 2019-10-24

[3] Abdelsadek, Y., Herrmann, F., Kacem, I., Otjacques, B.: Branch-and-bound algorithm for the maximum triangle packing problem. Computers & Industrial Engineering 81, 147–157 (March 2015)

[4] Agarwal, M., Srivastava, G.M.S.: A cuckoo search algorithm-based task scheduling in cloud computing. In: Bhatia, S.K., Mishra, K.K., Tiwari, S., Singh, V.K. (eds.) Advances in Computer and Computational Sciences. pp. 293–299. Springer Singapore, Singapore (2018)

[5] Anwar, N., Deng, H.: A hybrid metaheuristic for multi-objective scientific workflow scheduling in a cloud environment. Applied Sciences 8(4) (2018)

[6] Arellano, H., Tolentino, D., Gómez, R.: Optimum criss crossing cables in multi-span cable-stayed bridges using genetic algorithms. KSCE Journal of Civil Engineering 23(2), 719–728 (Feb 2019), https://doi.org/10.1007/s12205-018-5736-2

[7] Arora, S., Barak, B.: Computational Complexity: A Modern Approach, vol. 1. Cambridge University Press (April 2009)

[8] Bacanin, N., Tuba, M., Strumberger, I.: RFID network planning by ABC algorithm hybridized with heuristic for initial number and locations of readers. In: 2015 17th UKSim-AMSS International Conference on Modelling and Simulation (UKSim). pp. 39–44 (March 2015)

[9] Bacanin, N.: Implementation and performance of an object-oriented software system for cuckoo search algorithm. International Journal of Mathematics and Computers in Simulation 6(1), 185–193 (December 2010)

[10] Bacanin, N., Strumberger, I.: Klaud računarstvo. Singidunum Univerzitet (2018)

[11] Bacanin, N., Tuba, E., Bezdan, T., Strumberger, I., Tuba, M.: Artificial flora optimization algorithm for task scheduling in cloud computing environment. In: Yin, H., Camacho, D., Tino, P., Tallón-Ballesteros, A.J., Menezes, R., Allmendinger, R. (eds.) Intelligent Data Engineering and Automated Learning – IDEAL 2019. pp. 437–445. Springer International Publishing, Cham (2019), https://doi.org/10.1007/978-3-030-33607-3_47

[12] Bacanin, N., Tuba, M.: Artificial bee colony (ABC) algorithm for constrained optimization improved with genetic operators. Studies in Informatics and Control 21(2), 137–146 (June 2012)

[13] Bacanin, N., Tuba, M.: Firefly algorithm for cardinality constrained mean-variance portfolio optimization problem with entropy diversity constraint. The Scientific World Journal, special issue Computational Intelligence and Metaheuristic Algorithms with Applications, 2014(Article ID 721521), 16 (2014)

[14] Bacanin, N., Tuba, M.: Fireworks algorithm applied to constrained portfolio optimization problem. In: Proceedings of the 2015 IEEE Congress on Evolutionary Computation (CEC 2015) (May 2015)

[15] Bacanin, N., Tuba, M., Pelevic, B.: Krill herd (KH) algorithm for portfolio optimization. In: 14th International Conference on Mathematics and Computers in Business and Economics (MCBE '13). pp. 39–44 (2013)

[16] Bittencourt, L.F., Goldman, A., Madeira, E.R., da Fonseca, N.L., Sakellariou, R.: Scheduling in distributed systems: A cloud computing perspective. Computer Science Review 30, 31 – 54 (2018), http://www.sciencedirect.com/science/article/pii/S1574013718301187

[17] Boveiri, H.R., Khayami, R., Elhoseny, M., Gunasekaran, M.: An efficient swarm-intelligence approach for task scheduling in cloud-based internet of things applications. Journal of Ambient Intelligence and Humanized Computing (Oct 2018), https://doi.org/10.1007/s12652-018-1071-1

[18] Bozorgi, S.M., Yazdani, S.: Iwoa: An improved whale optimization algorithm for optimization problems. Journal of Computational Design and Engineering 6(3), 243 – 259 (2019), http://www.sciencedirect.com/science/article/pii/S2288430018301994

[19] Buyya, R., Pandey, S., Vecchiola, C.: Cloudbus toolkit for market-oriented cloud computing. In: Proceedings of the 1st International Conference on Cloud Computing. pp. 24–44. CloudCom '09, Springer-Verlag, Berlin, Heidelberg (2009), http://dx.doi.org/10.1007/978-3-642-10665-1_4

[20] C., E.R., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the 6th international symposium on micromachine and human science. pp. 39–43. IEEE (1995)

[21] Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A.F., Buyya, R.: Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. Pract. Exper. 41(1), 23–50 (January 2011), http://dx.doi.org/10.1002/spe.995

[22] Caramia, M., Dell'Olmo, P.: Multi-objective Management in Freight Logistics. No. 1 in ncreasing Capacity, Service Level and Safety with Optimization Algorithms, Springer-Verlag London (November 2008)

[23] Chandrasekaran, K.: Essentials of Cloud Computing. Chapman and Hall/CRC (2015)

[24] Chaudhary, D., Kumar, B.: Cloudy gsa for load scheduling in cloud computing. Applied Soft Computing 71, 861 – 871 (2018), http://www.sciencedirect.com/science/article/pii/S1568494618304344

[25] Chen, Y., Zhang, G.: Exchange rates determination based on genetic algorithms using mendel's principles: Investigation and estimation under uncertainty. Information Fusion 14(3), 327–333 (2013)

[26] Cheng, L., Qu, L., Xu, Y.: Artificial bee colony algorithm-based multiple-source localization method for wireless sensor network. In: 2017 2nd IEEE International Conference on Computational Intelligence and Applications (ICCIA). pp. 445–448 (Sep 2017)

[27] Cheraghalipour, A., Hajiaghaei-Keshteli, M., Paydar, M.M.: Tree growth algorithm (tga): A novel approach for solving optimization problems. Engineering Applications of Artificial Intelligence 72, 393 – 414 (2018), http://www.sciencedirect.com/science/article/pii/S0952197618301003

[28] Coello, C.A.C.: Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Information Fusion 191(11–12), 1245—-1287 (January 2002)

[29] Corne, D.W., Reynolds, A., Bonabeau, E.: Swarm Intelligence. Handbook of Natural Computing, Springer-Verlag (2012)

[30] Darvin, C.: On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life, First Edition. London: John Murray (1859)

[31] Darvis, Akay, B.: A modified artificial bee colony (ABC) algorithm for constrained optimization problems. Applied Soft Computing 11(3), 3021–3031 (2011)

[32] Das, S., Mullick, S.S., Suganthan, P.: Recent advances in differential evolution – an updated survey. Swarm and Evolutionary Computation 27, 1 – 30 (2016), http://www.sciencedirect.com/science/article/pii/S2210650216000146

[33] Deb, K.: An efficient constraint-handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering 186(2-4), 311–338 (2000)

[34] Elaziz, M.A., Xiong, S., Jayasena, K., Li, L.: Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution. Knowledge-Based Systems 169, 39 – 52 (2019), http://www.sciencedirect.com/science/article/pii/S0950705119300322

[35] Elhoseny, M., Abdelaziz, A., Salama, A.S., Riad, A., Muhammad, K., Sangaiah, A.K.: A hybrid model of internet of things and cloud computing to manage big data in health services applications. Future Generation Computer Systems 86, 1383 – 1394 (2018), http://www.sciencedirect.com/science/article/pii/S0167739X17322021

[36] Gao, R., Wu, J.: Dynamic load balancing strategy for cloud computing with ant colony optimizatio. Future Internet 7(4), 465–483 (2015)

[37] Ghosh, S., Kaur, M., Bhullar, S., Karar, V.: Hybrid abc-bat for solving short-term hydrothermal scheduling problems. Energies 12(3), 551 (2019)

[38] Glover, F.: Heuristics for integer programming using surrogate constraints. Decision Sciences 8(1), 156–166 (January 1977)

[39] Glover, F.: Future paths for integer programming and links to artificial intelligence. Computers & Operations Research 13(5), 533 – 549 (1986), http://www.sciencedirect.com/science/article/pii/0305054886900481, applications of Integer Programming

[40] Glover, F., McMillan, C.: The general employee scheduling problem: an integration of ms and ai. Computers and Operations Research, Special issue: Applications of integer programming 13(5), 563–573 (1986)

[41] Goldbogen, J.A., Friedlaender, A.S., Calambokidis, J., McKenna, M.F., Simon, M., Nowacek, D.P.: Integrative Approaches to the Study of Baleen Whale Diving Behavior, Feeding Performance, and Foraging Ecology. BioScience 63(2), 90–100 (02 2013), https://doi.org/10.1525/bio.2013.63.2.5

[42] Guerrero, C., Lera, I., Juiz, C.: Genetic algorithm for multi-objective optimization of container allocation in cloud architecture. Journal of Grid Computing 16(1), 113–135 (Mar 2018), https://doi.org/10.1007/s10723-017-9419-x

[43] Holland, J.H.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. Complex Adaptive Systems, A Bradford Book (1992)

[44] Hoopes, J.: Chapter 1 - an introduction to virtualization. In: Virtualization for Security, pp. 1 – 43. Syngress, Boston (2009), http://www.sciencedirect.com/science/article/pii/B9781597493055000013

[45] Jovanovic, R., Tuba, M.: An ant colony optimization algorithm with improved pheromone correction strategy for the minimum weight vertex cover problem. Applied Soft Computing 11(8), 5360–5366 (December 2011)

[46] Jungnickel, D.: Graphs, Networks and Algorithms, Algorithms and Computation in Mathematics, vol. 5. Springer-Verlag Berlin Heidelberg (2013)

[47] K., K., D., G.C., P., V.M.: Optimization by simulated annealing. Science Magazine 220(4598), 671–680 (1983)

[48] Kalra, M., Singh, S.: A review of metaheuristic scheduling techniques in cloud computing. Egyptian Informatics Journal 16(3), 275 – 295 (2015), http://www.sciencedirect.com/science/article/pii/S1110866515000353

[49] Kalra, M., Singh, S.: A review of metaheuristic scheduling techniques in cloud computing. Egyptian Informatics Journal 16(3), 275 – 295 (2015), http://www.sciencedirect.com/science/article/pii/S1110866515000353

[50] Karaboga, D.: An idea based on honey bee swarm for numerical optimization. Technical Report - TR06 pp. 1–10 (2005)

[51] Karaboga, D., Basturk, B.: Artificial bee colony (abc) optimization algorithm for solving constrained optimization problems. Foundations of Fuzzy Logic and Soft Computing, LNCS 4529, 789–798 (2007)

[52] Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. Journal of Global Optimization 39(3), 459–471 (april 2007)

[53] Karthikeyan, K., Sunder, R., Shankar, K., Lakshmanaprabu, S.K., Vijayakumar, V., Elhoseny, M., Manogaran, G.: Energy consumption analysis of virtual machine migration in cloud using hybrid swarm optimization (abc–ba). The Journal of Supercomputing (Sep 2018), https://doi.org/10.1007/s11227-018-2583-3

[54] Kaur, G., Kaur, K.: An adaptive firefly algorithm for load balancing in cloud computing. In: Deep, K., Bansal, J.C., Das, K.N., Lal, A.K., Garg, H., Nagar, A.K., Pant, M. (eds.) Proceedings of Sixth International Conference on Soft Computing for Problem Solving. pp. 63–72. Springer Singapore, Singapore (2017)

[55] Kaur, N., Singh, S.: A budget-constrained time and reliability optimization bat algorithm for scheduling workflow applications in clouds. Procedia Computer Science 98, 199 – 204 (2016), http://www.sciencedirect.com/science/article/pii/S1877050916321615, the 7th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2016)/The 6th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2016)/Affiliated Workshops

[56] Kaveh, A., Ghazaan, M.I.: Enhanced whale optimization algorithm for sizing optimization of skeletal structures. Mechanics Based Design of Structures and Machines 45(3), 345–362 (2017)

[57] Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks (ICNN '95). vol. 4, pp. 1942–1948 (1995)

[58] Keshanchi, B., Souri, A., Navimipour, N.J.: An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: Formal verification, simulation, and statistical testing. Journal of Systems and Software 124, 1 – 21 (2017), http://www.sciencedirect.com/science/article/pii/S0164121216301066

[59] ijaya Ketan Panigrahi, Shi, Y., Lim, M.H.: Handbook of Swarm Intelligence, Adaptation, Learning, and Optimization, vol. 8. Springer-Verlag (2011)

[60] Koza, J.R.: Genetic programming: On the Programming of Computers by Means of Natural Selection. MIT Press (December 1992)

[61] Koziel, S., Yang, X.S.: Computational Optimization, Methods and Algorithms, Studies in Computational Intelligence, vol. 356. Springer Berlin Heidelberg (2011)

[62] Krčevinac, S., Čangalović, M., Kovačević-Vujčić, V., Martić, M., Vujošević, M.: Operaciona istraživanja 1. FON (2006)

[63] Kumar, M., Sharma, S.: Pso-cogent: Cost and energy efficient scheduling in cloud environment with deadline constraint. Sustainable Computing: Informatics and Systems 19, 147 – 164 (2018), http://www.sciencedirect.com/science/article/pii/S2210537918300350

[64] Kumar, M., Sharma, S., Goel, A., Singh, S.: A comprehensive survey for scheduling techniques in cloud computing. Journal of Network and Computer Applications 143, 1 – 33 (2019), http://www.sciencedirect.com/science/article/pii/S1084804519302036

[65] Le, L.T., Nguyen, H., Dou, J., Zhou, J.: A comparative study of pso-ann, ga-ann, ica-ann, and abc-ann in estimating the heating load of buildings' energy efficiency for smart city planning. Applied Sciences 9(13), 2630 (2019)

[66] Li, J., Tian, Q., Zhang, G., Wu, W., Xue, D., Li, L., Wang, J., Chen, L.: Task scheduling algorithm based on fireworks algorithm. EURASIP Journal on Wireless Communications and Networking 2018(1), 256 (Oct 2018), https://doi.org/10.1186/s13638-018-1259-2

[67] Ling, Y., Zhou, Y., Luo, Q.: Lévy flight trajectory-based whale optimization algorithm for global optimization. IEEE Access 5, 6168–6186 (2017)

[68] Ling, Y., Zhou, Y., Luo, Q.: Lévy flight trajectory-based whale optimization algorithm for global optimization. IEEE Access 5, 6168–6186 (2017)

[69] Ma, L., Chen, H., Hu, K., Zhu, Y.: Hierarchical artificial bee colony algorithm for RFID network planning optimization. The Scientific World Journal 2014(Article ID 941532), 21 (2014)

[70] Mafarja, M.M., Mirjalili, S.: Hybrid whale optimization algorithm with simulated annealing for feature selection. Neurocomputing 260, 302 – 312 (2017), http://www.sciencedirect.com/science/article/pii/S092523121730807X

[71] Marco, D., M., G.L.: Ant colonies for the travelling salesman problem. Biosystems 43(2), 73–81 (July 1997)

[72] Marler, R.T., Arora, J.S.: The weighted sum method for multi-objective optimization: new insights. Structural and Multidisciplinary Optimization 41(6), 853–862 (June 2010)

[73] Martini, R., Reinelt, G.: The Linear Ordering Problem Exact and Heuristic Methods in Combinatorial Optimization, Applied Mathematical Sciences, vol. 175. Springer-Verlag Berlin Heidelberg (2011)

[74] McCall, J.: Genetic algorithms for modelling and optimisation. Journal of Computational and Applied Mathematics 184(1), 205 – 222 (2005), http://www.sciencedirect.com/science/article/pii/S0377042705000774, special Issue on Mathematics Applied to Immunology

[75] Mell, P., Grance, T.: The nist definition of cloud computing recommendations of the national institute of standards and technology. Gaithersburg, MD Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology 145 (2011)

[76] Meng, J., McCauley, S., Kaplan, F., Leung, V.J., Coskun, A.K.: Simulation and optimization of hpc job allocation for jointly reducing communication and cooling costs. Sustainable Computing: Informatics and Systems 6, 48 – 57 (2015), http://www.sciencedirect.com/science/article/pii/S2210537914000237, special Issue on Selected Papers from 2013 International Green Computing Conference (IGCC)

[77] Menon, G., Nabil, M., Narasimhan, S.: Branch and bound algorithm for optimal sensor network design. IFAC Proceedings Volumes 46(32), 690 – 695 (2013), http://www.sciencedirect.com/science/article/pii/S1474667015383385, 10th IFAC International Symposium on Dynamics and Control of Process Systems

[78] Mezura-Montes, E., Coello, C.A.C.: Constraint-handling in nature-inspired numerical optimization: Past, present and future. Swarm and Evolutionary Computation 1(4), 173–194 (December 2011)

[79] Michalewicz, Z., Schoenauer, M.: Evolutionary algorithms for constrained parameter optimization problems. Evolutionary Computation 4(1), 1–32 (1996)

[80] Mirjalili, S., Lewis, A.: The whale optimization algorithm. Advances in Engineering Software 95, 51 – 67 (2016), http://www.sciencedirect.com/science/article/pii/S0965997816300163

[81] Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. Advances in Engineering Software 69, 46 – 61 (2014), http://www.sciencedirect.com/science/article/pii/S0965997813001853

[82] Mishra, S.K., Sahoo, B., Parida, P.P.: Load balancing in cloud computing: A big picture. Journal of King Saud University - Computer and Information Sciences (2018), http://www.sciencedirect.com/science/article/pii/S1319157817303361

[83] Mladenovic, N., Hansen, P.: Variable neighborhood search. Journal of Computers and Operations Research 24(11), 1097–1100 (1997)

[84] Nagireddy, V., Parwekar, P., Mishra, T.K.: Velocity adaptation based pso for localization in wireless sensor networks. Evolutionary Intelligence (Sep 2018), https://doi.org/10.1007/s12065-018-0170-4

[85] Nocedal, J., Wright, S.: Numerical Optimization. No. 2 in Springer Series in Operations Research and Financial Engineering, Springer-Verlag New York (2006)

[86] Ogan, D., Azizoglu, M.: A branch and bound method for the line balancing problem in u-shaped assembly lines with equipment requirements. Journal of Manufacturing Systems 36, 46—-54 (July 2015)

[87] Palos-Sanchez, P.R.: Drivers and barriers of the cloud computing in smes: the position of the european union. Harvard Deusto Business Research VI(2), 116–132 (Oct 2017)

[88] Palos-Sanchez, P.R., Arenas-Marquez, F.J., Aguayo-Camacho, M.: Cloud computing (saas) adoption as a strategic technology: Results of an empirical study. Mobile Information Systems 2017, 20 (2017), http://www.sciencedirect.com/science/article/pii/S1877050917328600

[89] Parker, T.S., Chua, L.O.: Practical Numerical Algorithms for Chaotic Systems. Springer-Verlag, Berlin, Heidelberg (1989)

[90] Pelusi, D.: Optimization of a fuzzy logic controller using genetic algorithms. In: Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2011 International Conference on. vol. 2, pp. 143–146 (Aug 2011)

[91] Pinedo, M.: Scheduling: Theory, Algorithms, and Systems. Prentice Hall international series in industrial and systems engineering, Springer (2008), https://books.google.rs/books?id=EkpDak9kEs0C

[92] Prem Jacob, T., Pradeep, K.: A multi-objective optimal task scheduling in cloud environment using cuckoo particle swarm optimization. Wireless Personal Communications (May 2019), https://doi.org/10.1007/s11277-019-06566-w

[93] Przybylski, A., Gandibleux, X.: Multi-objective branch and bound. European Journal of Operational Research 260(3), 856 – 872 (2017), http://www.sciencedirect.com/science/article/pii/S037722171730067X

[94] Raghavan, S., Sarwesh, P., Marimuthu, C., Chandrasekaran, K.: Bat algorithm for scheduling workflow applications in cloud. In: 2015 International Conference on Electronic Design, Computer Networks Automated Verification (EDCAV). pp. 139–144 (Jan 2015)

[95] Rashedi, E., Nezamabadi-pour, H., Saryazdi, S.: Gsa: A gravitational search algorithm. Inf. Sci. 179(13), 2232–2248 (Jun 2009), https://doi.org/10.1016/j.ins.2009.03.004

[96] Rizk-Allah, R.M., Hassanien, A.E., Elhoseny, M., Gunasekaran, M.: A new binary salp swarm algorithm: development and application for optimization tasks. Neural Computing and Applications 31(5), 1641–1663 (May 2019), https://doi.org/10.1007/s00521-018-3613-z

[97] Rodriguez, F., Garcia-Martinez, C., Lozano, M.: Hybrid metaheuristics based on evolutionary algorithms and simulated annealing: taxonomy, comparison, and synergy test. IEEE Transactions on Evolutionary Computation 16(6), 787–800 (December 2012)

[98] Rothlauf, F.: Design of Modern Heuristics Principles and Application. No. 1 in Natural Computing Series, Springer-Verlag Berlin Heidelberg (February 2011)

[99] Shahi, B., Dahal, S., Mishra, A., Kumar, S.V., Kumar, C.P.: A review over genetic algorithm and application of wireless network systems. Procedia Computer Science 78, 431 – 438 (2016), http://www.sciencedirect.com/science/article/pii/S1877050916000879, 1st International Conference on Information Security & Privacy 2015

[100] Shi, Y.: Brain storm optimization algorithm. In: Tan, Y., Shi, Y., Chai, Y., Wang, G. (eds.) Advances in Swarm Intelligence. pp. 303–309. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)

[101] Singh, S.P., Sharma, S.C.: A pso based improved localization algorithm for wireless sensor network. Wireless Personal Communications 98(1), 487–503 (Jan 2018), https://doi.org/10.1007/s11277-017-4880-1

[102] Sreenu, K., Sreelatha, M.: W-scheduler: whale optimization for task scheduling in cloud computing. Cluster Computing (Jul 2017), https://doi.org/10.1007/s10586-017-1055-5

[103] Storn, R., Price, K.: Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Journal of Global Optimization 11, 341–359 (1997)

[104] Storn, R., Price, K.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11(4), 327–333 (December 1997)

[105] Strumberger, I., Bacanin, N., Tuba, M.: Constrained portfolio optimization by hybridized bat algorithm. In: 2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS). pp. 83–88 (Jan 2016)

[106] Strumberger, I., Tuba, E., Bacanin, N., Beko, M., Tuba, M.: Bare bones fireworks algorithm for the rfid network planning problem. In: 2018 IEEE Congress on Evolutionary Computation (CEC). pp. 1–8 (July 2018)

[107] Strumberger, I., Tuba, E., Bacanin, N., Beko, M., Tuba, M.: Hybridized moth search algorithm for constrained optimization problems. In: 2018 International Young Engineers Forum (YEF-ECE). pp. 1–5 (May 2018)

[108] Strumberger, I., Tuba, E., Bacanin, N., Beko, M., Tuba, M.: Modified monarch butterfly optimization algorithm for rfid network planning. In: 2018 6th International Conference on Multimedia Computing and Systems (ICMCS). pp. 1–6 (May 2018)

[109] Strumberger, I., Tuba, E., Bacanin, N., Beko, M., Tuba, M.: Wireless sensor network localization problem by hybridized moth search algorithm. In: 2018 14th International Wireless Communications Mobile Computing Conference (IWCMC). pp. 316–321 (June 2018)

[110] Strumberger, I., Tuba, E., Bacanin, N., Tuba, M.: Dynamic tree growth algorithm for load scheduling in cloud environments. In: 2019 IEEE Congress on Evolutionary Computation (CEC). pp. 65–72 (June 2019), https://doi.org/10.1109/CEC.2019.8790014

[111] Strumberger, I., Tuba, E., Bacanin, N., Zivkovic, M., Beko, M., Tuba, M.: Designing convolutional neural network architecture by the firefly algorithm. In: 2019 International Young Engineers Forum (YEF-ECE). pp. 59–65 (May 2019)

[112] Strumberger, I., Bacanin, N., Tuba, M.: Enhanced firefly algorithm for constrained numerical optimization, ieee congress on evolutionary computation. In: Proceedings of the IEEE International Congress on Evolutionary Computation (CEC 2017). pp. 2120–2127 (June 2017)

[113] Strumberger, I., Bacanin, N., Tuba, M.: Hybridized elephant herding optimization algorithm for constrained optimization. In: Abraham, A., Muhuri, P.K., Muda, A.K., Gandhi, N. (eds.) Hybrid Intelligent Systems. pp. 158–166. Springer International Publishing, Cham (2018)

[114] Strumberger, I., Bacanin, N., Tuba, M., Tuba, E.: Resource scheduling in cloud computing based on a hybridized whale optimization algorithm. Applied Sciences 9(22), 4893 (2019), https://doi.org/10.3390/app9224893

[115] Strumberger, I., Beko, M., Tuba, M., Minovic, M., Bacanin, N.: Elephant herding optimization algorithm for wireless sensor network localization problem. In: Camarinha-Matos, L.M., Adu-Kankam, K.O., Julashokri, M. (eds.) Technological Innovation for Resilient Systems. pp. 175–184. Springer International Publishing, Cham (2018)

[116] Strumberger, I., Minovic, M., Tuba, M., Bacanin, N.: Performance of elephant herding optimization and tree growth algorithm adapted for node localization in wireless sensor networks. Sensors 19(11), 2515 (2019), https://doi.org/10.3390/s19112515

[117] Strumberger, I., Sarac, M., Markovic, D., Bacanin, N.: Hybridized monarch butterfly algorithm for global optimization problems. International Journal of Computers 3 (April 2018)

[118] Strumberger, I., Sarac, M., Markovic, D., Bacanin, N.: Moth search algorithm for drone placement problem. International Journal of Computers 3 (April 2018)

[119] Strumberger, I., Tuba, E., Bacanin, N., Beko, M., Tuba, M.: Modified and hybridized monarch butterfly algorithms for multi-objective optimization. In: Madureira, A.M., Abraham, A., Gandhi, N., Varela, M.L. (eds.) Hybrid Intelligent Systems. pp. 449–458. Springer International Publishing, Cham (2020)

[120] Strumberger, I., Tuba, E., Zivkovic, M., Bacanin, N., Beko, M., Tuba, M.: Dynamic search tree growth algorithm for global optimization. In: Camarinha-Matos, L.M., Almeida, R., Oliveira, J. (eds.) Technological Innovation for Industry and Service Systems. pp. 143–153. Springer International Publishing, Cham (2019)

[121] Strumberger, I., Tuba, M., Bacanin, N., Tuba, E.: Cloudlet scheduling by hybridized monarch butterfly optimization algorithm. Journal of Sensor and Actuator Networks 8(3), 44 (2019), https://doi.org/10.3390/jsan8030044

[122] Sulaiman, N., Mohamad-Saleh, J., Abro, A.G.: A hybrid algorithm of abc variant and enhanced egs local search technique for enhanced optimization performance. Engineering Applications of Artificial Intelligence 74, 10 – 22 (2018), http://www.sciencedirect.com/science/article/pii/S0952197618301179

[123] SundarRajan, R., Vasudevan, V., Mithya, S.: Workflow scheduling in cloud computing environment using firefly algorithm. In: 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT). pp. 955–960 (March 2016)

[124] Talbi, E.G.: Metaheuristics: From Design to Implementation. No. 1, Wiley Publishing (July 2009)

[125] Tan, Y., Zhu, Y.: Fireworks algorithm for optimization. Advances in Swarm Intelligence, LNCS 6145, 355–364 (June 2010)

[126] T.O., T., Yang, X.S., Cheng, S., Huang, K.: Hybrid metaheuristic algorithms: Past, present, and future. Recent Advances in Swarm Intelligence and Evolutionary Computation Studies in Computational Intelligence 585, 71–83 (December 2015)

[127] Train, K.E., Wilson, W.W.: Monte carlo analysis of sp-off-rp data. Journal of Choice Modelling 2(1), 101–117 (2009)

[128] Tuba, E., Strumberger, I., Bacanin, N., Zivkovic, D., Tuba, M.: Cooperative clustering algorithm based on brain storm optimization and k-means. In: 2018 28th International Conference Radioelektronika (RADIOELEKTRONIKA). pp. 1–5 (April 2018)

[129] Tuba, E., Strumberger, I., Zivkovic, D., Bacanin, N., Tuba, M.: Mobile robot path planning by improved brain storm optimization algorithm. In: 2018 IEEE Congress on Evolutionary Computation (CEC). pp. 1–8 (July 2018)

[130] Tuba, E., Strumberger, I., Bacanin, N., Tuba, M.: Bare bones fireworks algorithm for capacitated p-median problem. In: Tan, Y., Shi, Y., Tang, Q. (eds.)

Advances in Swarm Intelligence. pp. 283–291. Springer International Publishing, Cham (2018)

[131] Tuba, E., Tuba, M., Dolicanin, E.: Adjusted fireworks algorithm applied to retinal image registration. Studies in Informatics and Control 26(1), 33–42 (March 2017)

[132] Tuba, M., Bacanin, N., Alihodzic, A.: Multilevel image thresholding by fireworks algorithm. In: 2015 25th International Conference Radioelektronika (RADIOELEKTRONIKA). pp. 326–330 (April 2015)

[133] Tuba, M., Bacanin, N., Beko, M.: Fireworks algorithm for rfid network planning problem. In: 2015 25th International Conference Radioelektronika (RADIOELEKTRONIKA). pp. 440–444 (April 2015)

[134] Tuba, M., Alihodzic, A., Bacanin, N.: Cuckoo Search and Bat Algorithm Applied to Training Feed-Forward Neural Networks, pp. 139–162. Springer International Publishing, Cham (2015), https://doi.org/10.1007/978-3-319-13826-8_8

[135] Tuba, M., Bacanin, N.: Artificial bee colony algorithm hybridized with firefly metaheuristic for cardinality constrained mean-variance portfolio problem. Applied Mathematics & Information Sciences 8(6), 2831–2844 (November 2014)

[136] Tuba, M., Bacanin, N.: Hybridized bat algorithm for multi-objective radio frequency identification (rfid) network planning. In: Proceedings of the 2015 IEEE Congress on Evolutionary Computation (CEC 2015) (May 2015)

[137] Tuba, M., Bacanin, N., Beko, M.: Multiobjective rfid network planning by artificial bee colony algorithm with genetic operators. In: Tan, Y., Shi, Y., Buarque, F., Gelbukh, A., Das, S., Engelbrecht, A. (eds.) Advances in Swarm and Computational Intelligence. pp. 247–254. Springer International Publishing, Cham (2015)

[138] Tuba, M., Bacanin, N., Pelevic, B.: Artificial bee colony algorithm for portfolio optimization problems. International Journal of Mathematical Models And Methods In Applied Sciences 7(10), 888–896 (2013)

[139] Tuba, M., Bacanin, N., Pelevic, B.: Framework for constrained portfolio selection by the firefly algorithm. International Journal of Mathematical Models and Methods in Applied Sciences 7(10), 1888–896 (2014)

[140] Tuba, M., Bacanin, N., Stanarevic, N.: Adjusted artificial bee colony (ABC) algorithm for engineering problems. WSEAS Transactions on Computers 11(4), 111–120 (2012)

[141] Tuba, M., Nebojsa: Improved seeker optimization algorithm hybridized with firefly algorithm for constrained optimization problems. Neurocomputing 143, 197–207 (2014)

[142] Wang, G.G.: Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems. Memetic Computing 10(2), 151–164 (Jun 2018), https://doi.org/10.1007/s12293-016-0212-3

[143] Wang, G.G., Deb, S., Cui, Z.: Monarch butterfly optimization. Neural Computing and Applications pp. 1–20 (May 2015)

[144] Wang, G.G., Deb, S., Gao, X.Z., dos S. Coelho, L.: A new metaheuristic optimisation algorithm motivated by elephant herding behaviour. International Journal of Bio-Inspired Computation 8(6), 394–409 (January 2017)

[145] Wang, G.G., Deb, S., dos S. Coelho, L.: Elephant herding optimization. In: Proceedings of the 2015 3rd International Symposium on Computational and Business Intelligence (ISCBI). pp. 1–5 (December 2015)

[146] Wang, G.G., Gandomi, A.H., Alavi, A.H.: An effective krill herd algorithm with migration operator in biogeography-based optimization. Applied Mathematical Modelling 38(9–10), 2454—-2462 (May 2014)

[147] Wang, T., Liu, Z., Chen, Y., Xu, Y., Dai, X.: Load balancing task scheduling based on genetic algorithm in cloud computing. In: 2014 IEEE 12th Internatio-

nal Conference on Dependable, Autonomic and Secure Computing. pp. 146–152 (Aug 2014)

[148] Weise, T.: Global optimization algorithms – theory and application (2008)

[149] Xin Yao, Yong Liu, Guangming Lin: Evolutionary programming made faster. IEEE Transactions on Evolutionary Computation 3(2), 82–102 (July 1999)

[150] Xu, B., Sun, Z.: A fuzzy operator based bat algorithm for cloud service composition. Int. J. Wire. Mob. Comput. 11(1), 42–46 (January 2016), https://doi.org/10.1504/IJWMC.2016.079471

[151] Yakhchi, M., Ghafari, S.M., Yakhchi, S., Fazeli, M., Patooghi, A.: Proposing a load balancing method based on cuckoo optimization algorithm for energy management in cloud computing infrastructures. In: 2015 6th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO). pp. 1–5 (May 2015)

[152] Yang, X.S.: Firefly algorithms for multimodal optimization. Stochastic Algorithms: Foundations and Applications, LNCS 5792, 169–178 (2009)

[153] Yang, X.S.: Firefly algorithm, stochastic test functions and design optimisation. International Journal of Bio-Inspired Computation 2(2), 78–84 (2010)

[154] Yang, X.S.: Nature-Inspired Metaheuristic Algorithms: Second Edition. Luniver Press (July 2010)

[155] Yang, X.S.: A new metaheuristic bat-inspired algorithm. Studies in Computational Intelligence 284, 65—74 (November 2010)

[156] Yang, X.S.: Optimization Algorithms, pp. 13–31. Springer Berlin Heidelberg, Berlin, Heidelberg (2011), https://doi.org/10.1007/978-3-642-20859-1_2

[157] Yang, X.S.: Chapter 1 - introduction to algorithms. In: Yang, X.S. (ed.) Nature-Inspired Optimization Algorithms, pp. 1 – 21. Elsevier, Oxford (2014), http://www.sciencedirect.com/science/article/pii/B9780124167438000014

[158] Yang, X.S.: Swarm intelligence based algorithms: A critical analysis. Evolutionary Intelligence 7(1), 17–28 (April 2014)

[159] Yang, X.S., Deb, S.: Cuckoo search via levy flights. In: Proceedings of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009). pp. 210–214 (2009)

[160] Yang, X.S., Karamanoglu, M.: 1 - swarm intelligence and bio-inspired computation: An overview. In: Yang, X.S., Cui, Z., Xiao, R., Gandomi, A.H., Karamanoglu, M. (eds.) Swarm Intelligence and Bio-Inspired Computation, pp. 3 – 23. Elsevier, Oxford (2013), http://www.sciencedirect.com/science/article/pii/B9780124051638000016

[161] Yang, X.S., Ting, T.O., Karamanoglu, M.: Random walks, lévy flights, markov chains and metaheuristic optimization. Future Information Communication Technology and Applications Lecture Notes in Electrical Engineering 235(3), 1055–1064 (May 2013)

[162] Yokoyama, R., Shinano, Y., Taniguchi, S., Ohkura, M., Wakui, T.: Optimization of energy supply systems by milp branch and bound method in consideration of hierarchical relationship between design and operation. Energy Conversion and Management 92, 92—-104 (March 2015)

[163] Zahoor, S., Javaid, S., Javaid, N., Ashraf, M., Ishmanov, F., Afzal, M.K.: Cloud–fog–based smart grid model for efficient resource management. Sustainability 10(6), 2079 (September 2016)

[164] Zhang, L., Tang, Y., Hua, C., Guan, X.: A new particle swarm optimization algorithm with adaptive inertia weight based on bayesian techniques. Applied Soft Computing 28, 138–149 (June 2015)

[165] Zuo, L., Shu, L., Dong, S., Zhu, C., Hara, T.: A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing. IEEE Access 3, 2687–2699 (2015)

UNAPREĐENJE RASPOREĐIVANJA POSLOVA I BALANSIRANJA OPTEREĆENJA U KLAUD OKRUŽENJU PRIMENOM METAHEURISTIKA INTELIGENCIJE ROJEVA

ORIGINALITY REPORT

2%

SIMILARITY INDEX

PRIMARY SOURCES

1	www.mdpi.com Internet	323 words — 1%
2	nardus.mpn.gov.rs Internet	305 words — 1%
3	link.springer.com Internet	67 words — < 1%
4	de.scribd.com Internet	47 words — < 1%
5	dspace.iimk.ac.in Internet	40 words — < 1%
6	res.mdpi.com Internet	31 words — < 1%
7	www.cis.hr Internet	28 words — < 1%
8	www.algebra.hr Internet	22 words — < 1%
9	nrl.northumbria.ac.uk Internet	13 words — < 1%
10	Ivana Strumberger, Milan Tuba, Nebojsa Bacanin, Eva Tuba. "Cloudlet Scheduling by Hybridized	12 words — < 1%

Monarch Butterfly Optimization Algorithm", Journal of Sensor and Actuator Networks, 2019
Crossref

11	"Hybrid Intelligent Systems", Springer Science and Business Media LLC, 2020 Crossref	12 words —	< 1%
12	"Information and Communication Technology for Intelligent Systems", Springer Nature, 2019 Crossref	11 words —	< 1%
13	fbe.firat.edu.tr Internet	11 words —	< 1%
14	www.scribd.com Internet	11 words —	< 1%
15	www.jpier.org Internet	10 words —	< 1%
16	citeseerx.ist.psu.edu Internet	10 words —	< 1%

EXCLUDE QUOTES ON
EXCLUDE BIBLIOGRAPHY ON
EXCLUDE MATCHES < 15 WORDS