DRŽAVNI UNIVERZITET U NOVOM PAZARU

DEPARTMAN ZA TEHNIČKE NAUKE

Irfan S. Fetahović

UNAPREĐENJE PLANIRANJA PUTANJE ROBOTA BRAIN STORM ALGORITMOM

doktorska disertacija

Novi Pazar, 2018

STATE UNIVERSITY OF NOVI PAZAR

DEPARTMENT OF TECHNICAL SCIENCES

Irfan S. Fetahović

IMPROVEMENT OF ROBOT PATH PLANNING BY BRAIN STORM ALGORITHM

Doctoral Dissertation

Novi Pazar, 2018

Podaci o komisiji

Mentori:

prof. dr Milan Tuba, redovni profesor Državnog univerziteta u Novom Pazaru

prof. dr Edin Dolićanin, vanredni profesor Državnog univerziteta u Novom Pazaru

Članovi komisije:

prof. dr Boško Nikolić, redovni profesor Elektrotehničkog fakulteta Univerziteta u Beogradu

prof. dr Milan Tuba, redovni profesor Državnog univerziteta u Novom Pazaru

prof. dr Edin Dolićanin, vanredni profesor Državnog univerziteta u Novom Pazaru

prof. dr Yuhui Shi, redovni profesor, Southern University of Science and Technology, Šenžen, Kina

Datum odbrane:

Podaci o doktorskoj disertaciji

Naslov doktorske disertacije: Unapređenje planiranja putanje robota brain storm algoritmom

Rezime

Pod planiranjem putanje se podrazumeva nalaženje optimalne putanje od početne do ciljne tačke, pri čemu se optimalnost može definisati na različite načine. Primenjuje se na planiranje putanje različitih objekata u različitim okruženjima pod različitim uslovima. Pod robotom se ovde u najširem smislu podrazumeva bilo kakav pokretni objekat, ali se pod planiranjem putanje može podrazumevati i planiranje kretanja raznih veštačkih ruku kakve se koriste u industriji. Prilikom planiranja putanje moraju se uzeti u obzir brojni zahtevi i ograničenja. Putanja kojom se robot kreće treba da bude najkraća moguća tako da potrošnja goriva bude minimalna, kao i da se od startne do ciljne pozicije stigne za najkraće vreme. Osim toga, zahteva se da ne dolazi do kolizije robota sa preprekama ili sa drugim robotima, a u slučaju UAV letelica potrebno je minimizovati izlaganje UAV pretnjama u vidu neprijateljskih radara, raketa ili drugih letelica. Dodatna ograničenja UAV letelice koja moraju biti razmatrana prilikom planiranja putanje su: ograničenja ugla okretanja, ograničenja ugla uspona/poniranja, visina leta, i ugao prilaska ciljnoj poziciji. Planiranje putanje se može razumeti kao višeciljni optimizacioni problem sa ograničenjima koji uključuje različite ciljeve, kao i brojne zahteve i ograničenja. Nažalost, ovi ciljevi i zahtevi su često međusobno protivrečni, te se prilikom nalaženja optimalne putanje mora praviti određeni kompromis. Problem planiranja putanje je težak optimizacioni problem tako da ne postoje determinističke metode za njegovo rešavanje u razumnom vremenu. Iako je ovaj problem već rešavan primenom različitih metoda i tehnika, primena algoritama inteligencije rojeva na rešavanje ovog problema nije dovoljno izučavana. Brain storm optimizacioni algoritam je algoritam inteligencije rojeva koji je inspirisan procesom generisanja ideja od strane ljudi prilikom rešavanja nekog problema, tj. na brainstorming procesu. Razvio ga je Yuhui Shi 2011. godine i od tada je uspešno korišćen za rešavanje različitih optimizacionih problema.

U ovoj disertaciji smo analizirali načine rešavanja problema planiranja putanje

robota primenom prirodom inspirisanih metaheuristika, posebno posvećujući pažnju algoritmima inteligencije rojeva. Prilagodili smo i primenili brain storm optimizacioni algoritam na rešavanje problema planiranja putanje robota.

U prvom delu istraživanja smo primenili osnovnu verziju BSO algoritma na problem planiranje putanje UCAV letelica. Potrošnja goriva i bezbednost su razmatrani kao kriterijumi optimalnosti putanje. Predloženi metod je testiran u test okruženju iz literature koji podrazumeva kružno predstavljanje zona opasnosti i različite nivoe pretnje. Naš predloženi metod je bio upoređivan sa deset drugih, prirodom inspirisanih metaheuristika iz literature. Na osnovu rezultata simulacije, može se zaključiti da je predloženi brain storm optimizacioni algoritam robustan, da je pokazao bolje performanse u skoro svim slučajevima i da ima potencijal za dalja poboljšanja. On je pokazao bolje performanse za manje dimenzije problema, dok je za veće dimenzije bilo potrebno više iteracija, mada su rezultati i u tom slučaju pokazali poboljšanje u poređenju sa ostalim testiranim algoritmima.

U drugom delu istraživanja smo primenili osnovni BSO na problem planiranja putanje robota u neizvesnom okruženju sa statičkim preprekama. Predložili smo probabilistički model za određivanje stepena opasnosti, za izvore opasnosti sa nepoznatim tačnim pozicijama. Dva kontradiktorna kriterijuma, dužina putanje i bezbednost, su povezana uvođenjem kontrolnog parametra u funkciju cilja. Predloženi metod rešava problem neizvodljivih rešenja dodavanjem vrednosti kazne u funkciju cilja. Kombinacija kazne i povećane eksploracije je obezbedila da se uvek generišu izvodljiva rešenja. Poređenjem sa konkurentnim algoritmom je dokazano da je naš predloženi metod, iako jednostavniji, efikasniji i robustniji zato što su dobijena bolja rešenja u svim slučajevima.

Na kraju je razmatran problem planiranja putanje mobilnog robota u dvodimenzionalnom okruženju primenom poboljšanog BSO algoritma. Predloženi metod kombinuje BSO algoritam sa metodom lokalnog pretraživanja da bi se našla najkraća putanja u grafu, u cilju pretraživanja optimalne putanje u okruženju sa statičkim preprekama. Dužina putanje je korišćena kao jedini cilj. Inicijalna izvodljiva rešenja za BSO su generisana primenom determinističke procedure lokalnog pretraživanja, a BSO je korišćen da bi se dalje optimizovala putanja. Predloženi metod je testiran za pet različitih scenarija i dokazano je da može da nađe optimalna izvodljiva rešenja.

Buduća istraživanja mogu uključivati: hibridizaciju i modifikaciju brain storm optimizacionog algoritma kako bi se poboljšala brzina konvergencije i algoritam prilagodio rešavanju problema sa većim dimenzijama; analizu problema planiranja putanje u 3D okruženju; i dizajniranje algoritama za samoadaptivno i kolaborativno planiranje putanje. S obzirom da putanje nisu samo skupovi tačaka, već imaju brojne inherentne relacije i zavisnosti, tu činjenicu je moguće iskoristiti kako bi se napravili efikasniji algoritmi. U daljem istraživanju, umesto modela okruženja baziranog na gridu, mogu se koristiti realni prostori pretraživanja, a inicijalne tačke se mogu dobiti korišćenjem određenih smernica umesto korišćenjem tačaka slučajno raspoređenih u prostoru pretraživanja.

Ključne reči: planiranje putanje robota, brain storm algoritam, inteligencija rojeva, prirodom inspirisane metaheuristike
Naučna oblast: Računarske nauke
Uža naučna oblast: Veštačka inteligencija
UDK broj:

Dissertation Data

Doctoral Dissertation Title: Improvement of robot path planning by brain storm algorithm

Abstract

Path planning can be defined as finding optimal route between start and target point, where optimality can be defined in numerous ways. It can be applied to path planning of different objects in various environments under different conditions. Here, the robot is most widely understood as any moving object, but path planning can also refer to motion planning of different artificial hands that are used in the industry. During path planning process many demands and constraints must be taken into consideration. The path along which the robot moves must be the shortest possible so the fuel consumption remains at minimum, and the time needed to reach the target point is minimized. Moreover, the collisions between robot and obstacles or other robots must be avoided, and in the case of UAV vehicles, the exposure of these vehicles to threats, such as radars, missiles or hostile aircrafts, also must be minimized. In addition, path planning for UAVs has even more constraints which must be taken into consideration such as: turning angle, climbing/diving angle, flight altitude etc. Path planning can be understood as a multi-objective constrained optimization problem which includes different objectives, as well as numerous demands and constraints. Unfortunately, these objectives and constraints are often mutually exclusive, thus compromises must be made during path planning process. Path planning problem is a hard optimization problem so there is no successful deterministic method to solve it in reasonable time. Although different methods and techniques were proposed to solving path planning problem, the application of swarm intelligence algorithms to robot path planning is not sufficiently researched. Brain storm optimization algorithm is swarm intelligence algorithm inspired by the human idea generation process during problem solving, i.e. brainstorming process. It was designed by Yuhui Shi in 2011. and since it was successfully applied to solving numerous optimization problems.

In this dissertation, we analysed methods of solving robot path planning problem using nature-inspired algorithms, especially swarm intelligence algorithms. Moreover,

we adjusted and applied brain storm optimization algorithm to robot path planning.

In the first part of our research we applied the original brain storm algorithm for UCAV path planning. Fuel consumption and safety were considered as performance criteria. The proposed method was tested in the environment from the literature, with circular danger zones and different threat degrees. Our proposed method was compared with ten other, nature-inspired metaheuristics from the literature. Based on the simulation results, it can be concluded that our proposed approach is robust, exhibits better performance in almost all cases and has potential for further improvements. It had better performance for smaller problem dimensions, while for larger problem dimensions more iterations were needed. However, the results were further improved compared to other algorithms.

In the second part of the research we applied the original BSO for robot path planning in uncertain environment with static obstacles. We proposed probabilistic model for determining danger degree for sources with unknown certain positions. Two contradicted criteria, path length and safety, are handled by introducing controlling parameter into the objective function. The proposed method deals with infeasible solution by adding penalty to the objective function. Penalty combined with increased exploration was capable to ensure that feasible solutions are always generated. Comparison with concurrent algorithm proves that our proposed method is, even though simpler, more efficient and robust since it obtained best solutions in all cases.

Finally, we considered mobile robot robot path planning problem in two dimensional grid based space. Brain storm optimization combined with the local search method for finding the shortest path in the graph was used for searching the optimal path in environments with static obstacles. Only path length was used as objective. Initial feasible solution for the BSO were generated by local search deterministic procedure and the BSO was used to further optimize the path. The proposed method was tested in five different scenarios and proved to be able to find the optimal feasible solution.

A future work can include: hybridization or modification of the brain storm optimization algorithm in order to improve the convergence speed and to adjust it for larger dimensional problems; analysis of path planning problem in 3D environment; and designing algorithms for self-adaptive and collaborative path planning. Future research can introduce numerous improvements. Since paths are far from random col-

lections of points, with many inherent relations and dependencies, it may be possible to exploit that and guide path formation in order to make more efficient algorithms. In further research, instead of grid based environment model, real search space can be used and the initial points can be obtained by some guidance instead of using randomly deployed points in the search space.

Keywords: robot path planning, brain storm algorithm, swarm intelligence, nature-inspired metaheuristics
Scientific field: Computer science
Scientific discipline: Artificial intelligence
UDC number:

Predgovor

Planiranje putanje robota, odnosno, određivanje optimalne putanje robota od početne do ciljne tačke je optimizacioni problem od velikog praktičnog značaja. Primenjuje se na planiranje putanje različitih objekata u različitim okruženjima pod različitim uslovima. Neki od primera su planiranje putanje u dve ili tri dimenzije podvodnih vozila, pokretnih senzora, različitih bespilotnih letelica, te planiranje putanje robota u radioaktivnom okruženju, nepoznatim i neizvesnim okruženjima, kao i okruženjima sa velikim brojem prepreka. Pod robotom se ovde u najširem smislu podrazumeva bilo kakav pokretni objekat, ali se pod planiranjem putanje može podrazumevati i planiranje kretanja raznih veštačkih ruku kakve se koriste u industriji.

Na osnovu objavljenih naučnih radova, može se zaključiti da je problem planiranja putanje veoma aktuelan, i da je rešavan primenom velikog broja tehnika, metoda i metaheuristika. S druge strane, vidi se i da je ovaj problem nedovoljno rešavan primenom metaheuristika inteligencije rojeva (swarm intelligence) koje spadaju u grupu metaheuristika inspirisanih prirodom (nature-inspired). S obzirom na činjenicu da je problem planiranja putanje problem globalne optimizacije, a da algoritmi inteligencije rojeva postižu odlične rezultate u rešavanju problema globalne optimizacije, potrebno je ispitati potencijal algoritama inteligencije rojeva u rešavanju navedenog problema. Yuhui Shi, uz Kennedy-ja i Eberhart-a, treći glavni osnivač PSO algoritma, nedavno je predložio novi brain storm optimizacioni algoritam koji je vrlo perspektivan, ali za sada nedovoljno izučen.

Cilj ove doktorske disertacije je analiza mogućnosti i ispitivanje performansi algoritama inteligencije rojeva na rešavanje problema planiranja putanje robota, sa posebnim naglaskom na brain storm optimizacioni algoritam, odnosno, na mogućnost unapređenja planiranja putanje robota primenom brain storm optimizacionog algoritma.

Disertacija "Unapređenje planiranja putanje robota brain storm algoritmom"je napisana na 144 strane, formata A4 u LaTeX-u, i sadrži 17 slika, 12 tabela i 29 algoritama. Pored predgovora, ova doktorska disertacija sadrži 6 poglavlja i zaključak. Na kraju su dati spisak korišćene literature i kratka biografija autora.

U prvom poglavlju je data definicija optimizacije kao naučne discipline. Opisane su vrste optimizacionih problema i metoda za njihovo rešavanje, sa posebnim osvrtom

na prirodom inspirisane metaheuristike. Prikazan je kratak opis algoritma simuliranog kaljenja, kao tipičnog predstavnika metoda trajektorije, a genetski algoritam i diferencijalna evolucija su opisani kao predstavnici populacionih metoda.

Drugo poglavlje se bavi posebnom klasom prirodom inspirisanih algoritama koji se nazivaju algoritmi inteligencije rojeva. Najpre su date opšte osobine ovih metoda, a zatim je opisano nekoliko najznačajnijih algoritama koji pripadaju ovoj klasi, i to: algoritam kolonije mrava, optimizacija rojevima čestica, algoritam veštačke kolonije pčela, algoritam svica, kukavičje pretraživanje i algoritam slepog miša.

Brain storm optimizacioni algoritam, koji pripada algoritmima inteligencije rojeva, je detaljno opisan u poglavlju 3. Data je osnovna verzija ovog algoritma, kao i brojne modifikacije i hibridizacije. Takođe, prikazane su i varijante ovog algoritma za rešavanje višeciljnih i multimodalnih problema. Poglavlje sadrži i analizu mogućnosti BSO algoritma za rešavanje različitih optimizacionih problema, kao i ispitivanje uticaja parametara algoritma na njegove performanse. Na kraju poglavlja su date najznačajnije primene BSO algoritma.

Poglavlje 4 sadrži opis problema planiranja putanje robota. Na početku su date definicije osnovnih pojmova, te razmatranje osobina i ograničenja putanje. Nakon toga, predstavljeni su tradicionalni pristupi i metode za planiranje putanje. Na kraju poglavlja se razmatra okvir za rešavanje ovog problema primenom optimizacionih algoritama.

Pregled rešenja za problem planiranja putanje robota primenom prirodom inspirisanih metaheuristika je dat u poglavlju 5. Opisana su rešenja koja pripadaju svim najznačajnijim predstavnicima ove klase metoda, posebno posvećujući pažnju algoritmima inteligencije rojeva.

Poglavlje 6 sadrži predlog rešenja za planiranje putanje robota primenom brain storm optimizacionog algoritma. Osnovna verzija BSO algoritma je primenjena na problem planiranja putanje UCAV letelica, kao i na određivanje optimalne putanje mobilnog robota u neizvesnom okruženju sa statičkim preprekama i dinamičkim stohastičkim izvorima opasnosti. Dodatno, predložena je poboljšana verzija BSO algoritma, koji koristi proceduru lokalnog pretraživanja, radi rešavanja problema putanje mobilnog robota u dvodimenzionalnom okruženju.

* * * * * * * * *

Zahvaljujem se svojim mentorima, prof. dr Milanu Tubi i prof. dr Edinu Dolićaninu na pomoći i savetima tokom izrade ove doktorske disertacije i dosadašnjeg naučnog rada. Takođe se zahvaljujem prof. dr Bošku Nikoliću na komentarima i korisnim sugestijama. Posebnu zahvalnost dugujem svojoj porodici, ocu Salihu, majci Magbuli, sestri Azri i bratu Feridu na stalnoj i bezrezervnoj podršci koja mi je dala motivaciju i snagu da istrajem na putu nauke i znanja.

Sadržaj

1 OPTIMIZACIJA

Većina resursa koje koristimo u svakodnevnom životu (npr. vreme, novac i sl.) su ograničeni. U mnogim aktivnostima koje obavljamo mi se trudimo da optimizujemo te ograničene resurse kako bismo ih koristili na najbolji mogući način. Pod optimizacijom ovde podrazumevamo nalaženje najboljeg rešenja u datom kontekstu koji čine mogući skup rešenja, definisani ciljevi i ograničenja [1]. Dakle, optimizacija kao aktivnost je svuda oko nas; koristi se u ekonomiji, računarskim mrežama, energetici, transportu, ali i u svakodnevnom životu, za planiranje izleta ili privatnog putovanja. Optimizacija je od ključne važnosti u svim oblastima i aktivnostima koje uključuju donošenje odluka jer je ono direktno povezano sa postupkom biranja između više mogućih alternativa. Izbor jedne (ili više) od mogućih alternativa tj. rešenja problema je određeno našom željom da donesemo najbolju moguću odluku, pri čemu je mera kvaliteta odluke definisana ciljem koji želimo da postignemo, odnosno, ako govorimo matematičkim jezikom, funkcijom cilja.

Većinu optimizacionih problema je, matematički gledano, moguće opisati na sledeći način [2]:

$$\underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimizuj}} \quad f_i(\mathbf{x}), \qquad (i = 1, 2, \ldots, M), \tag{1.1}$$

$$\text{ako važi} \qquad h_j(\mathbf{x}) = 0, \qquad (j = 1, 2, \ldots, J), \tag{1.2}$$

$$g_k(\mathbf{x}) \leq 0, \qquad (k = 1, 2, \ldots, K), \tag{1.3}$$

gde su $f_i(\mathbf{x})$, $h_j(\mathbf{x})$, i $g_k(\mathbf{x})$ funkcije vektora dizajna

$$\mathbf{x} = (x_1, x_2, ..., x_n)^T \tag{1.4}$$

Vektor $\mathbf{x}$ iz (1.4) u stvari predstavlja rešenje optimizacionog problema, a njegove komponente x_i se nazivaju promenljive odlučivanja (decision variables) ili promenljive dizajna (design variables). Funkcija $f_i(\mathbf{x})$ se naziva funkcija cilja (objective function), fitnes funkcija, kriterijumska funkcija ili mera performansi. Zadatak optimizacije je nalaženje onog rešenja koje daje minimalnu (zadatak minimizacije) ili maksimalnu (zadatak maksimizacije) vrednost funkcija cilja, kao i samu vrednost kriterijumskih

funkcija u tom slučaju. Jednačina (1.1) je napisana za slučaj problema minimizacije. Prostor koji definišu vrednosti promenljivih odlučivanja se naziva prostor pretraživanja (search space) ili prostor dizajna (design space) $\mathbb{R}^n$, a prostor koji je definisan vrednostima funkcija cilja se naziva prostor rešenja (solution space). Funkcije $h_j(\mathbf{x})$ i $g_k(\mathbf{x})$ predstavljaju ograničenja, pri čemu je funkcije $g_k(\mathbf{x})$ moguće definisati i kao $g_k(\mathbf{x}) \geq 0$.

1.1 Klasifikacija optimizacionih problema

Optimizacioni problemi se mogu klasifikovati po raznim kriterijumima. U zavisnosti od broja ciljeva optimizacija može biti jednociljna (single-objective), ako je $M = 1$, ili višeciljna (multi-objective), ako je $M > 1$. Većina realnih problema su višeciljni, i za njih je poznato da su složeniji i teži za rešavanje. Ako je u jednačinama (1.2) i (1.3) $J = 0$ i $K = 0$, onda je u pitanju optimizacioni problem bez ograničenja (unconstrained optimization problem), u suprotnom, u pitanju je optimizacioni problem sa ograničenjima, pri čemu za $J = 0$ i $K \geq 1$ govorimo o optimizacionom problemu ograničenom nejednakostima (inequality-constrained problem), a ako je $K = 0$ i $J \geq 1$ onda je u pitanju optimizacioni problem ograničen jednakostima (equality-constrained problem). Deo prostora pretraživanja za koji važi da su sva ograničenja zadovoljena se naziva dopustivi prostor (feasible space). U slučaju problema ograničene optimizacije, pretraga za optimalnim rešenjem ima smisla samo unutar ovog prostora.

Klasifikacija se može obaviti i na osnovu oblika funkcija cilja i funkcija koje definišu ograničenja. Ukoliko su sve funkcije $h_j(\mathbf{x})$ i $g_k(\mathbf{x})$ linearne, onda je reč o linearno ograničenom problemu. Dodatno, ako su pritom i funkcije cilja, $f_i(\mathbf{x})$ linearne, onda je reč o problemu linearnog programiranja. Kada govorimo o klasifikaciji na osnovu oblika funkcija cilja i ograničenja važno je pomenuti i sledeće vrste optimizacije: kvadratno programiranje (funkcije cilja su kvadratne funkcije, a funkcije ograničenja su linearne), nelinearno programiranje (funkcije cilja i ograničenja su nelinearne), geometrijsko programiranje (funkcije cilja i ograničenja su pozinomi, tj. polinomi sa pozitivnim koeficijentima), konveksna optimizacija (funkcije cilja su konveksne), ne-glatka optimizacija (funkcije cilja i/ili ograničenja nisu diferencijabilne). Lokalna optimizacija je ona gde se pretraga vrši u blizini optimalne tačke, dok globalna optimizacija uzima o obzir ceo domen funkcije cilja [3].

Klasifikacija se može izvršiti i na osnovu tipa promenljivih. Ako su promenljive

kontinualne reč je o istoimenoj optimizaciji, a u slučaju diskretnih promenljivih (npr. binarne ili celobrojne) govorimo o diskretnoj optimizaciji. Dve poznate grane diskretne optimizacije su celobrojno programiranje i kombinatorna optimizacija. Celobrojno programiranje se odnosi na probleme u kojima sve promenljive pripadaju skupu celih brojeva. U slučaju da se umesto celobrojnih ili binarnih promenljivih koriste diskretne strukture kao što su grafovi, matroidi i sl. tada govorimo o kombinatornoj optimizaciji. Ove dve grane su međusobno povezane jer se mnogi kombinatorni problemi mogu modelovati kao celobrojno programiranje [4]. Drugim rečima, diskretna ili kombinatorna optimizacija se odnose na problem pretraživanja radi nalaženja optimalnog rešenja u konačnom ili beskonačno prebrojivom skupu potencijalnih rešenja. Rešenja ovakvih problema mogu biti kombinatorne strukture poput sekvenci, kombinacija, podskupova, podgrafova i sl. Da bi se kod ovakvih problema definisao koncept lokalnog minimuma najpre treba odrediti pogodnu metriku za definisanje rastojanja između rešenja, odnosno za definisanje koncepta susedstva u prostoru rešenja [5]. Važno je istaći da su ove metrike zavisne od problema koji se rešava.

Kod determinističke optimizacije parametri u optimizacionom problemu (promenljive ili funkcije) su unapred poznati, a postupak sledi ponovljivu i matematički rigoroznu proceduru koja isključuje bilo kakvu slučajnu prirodu. Kod takve optimizacije, ako se krene od jedne iste početne tačke, uvek će se doći do istog rešenja. Kada su parametri sistema slučajne prirode (zbog prisustva slučajnog šuma i sl.) ili ako su metode rešavanja problema takve da se prilikom pretrage prave slučajni izbori, reč je o stohastičkoj optimizaciji. Za razliku od unimodalnih problema koji imaju jedno rešenje, multimodalni optimizacioni problemi se mogu definisati kao oni problemi koji imaju više globalnih ili lokalnih optimuma, tj. optimalnih rešenja. Za ovu vrstu problema interesantno je dobiti najveći mogući broj rešenja iz nekoliko razloga; s jedne strane, kad ne postoji kompletno znanje o problemu, dobijeno rešenje ne mora biti i najbolje jer se ne može tvrditi da bolje rešenje ne može biti nađeno u prostoru pretraživanja koji nije u potpunosti istražen; s druge strane, iako smo sigurni da je dobijeno najbolje rešenje, takođe mogu postojati i druga, jednako dobra ili čak malo lošija rešenja koja se mogu preferirati iz različitih razloga (jednostavnija interpretacija, lakša implementacija i sl), te se ova rešenja mogu smatrati globalno boljim [6].

U višekriterijumskoj (višeciljnoj) optimizaciji se veoma često dešava da ne postoji

rešenje koje je najbolje po svim kriterijumima, odnosno, jedno rešenje je optimalno za jednu kriterijumsku funkciju, ali nije za drugu. Zato se uvodi koncept za ocenu optimalnosti rešenja koji se naziva Pareto optimalnost. Dopustivo rešenje $x^* \in X$ je Pareto optimalno (efikasno, dominirajuće, nedominirano) ako ne postoji drugo rešenje $x \in X$ koje zadovoljava oba od sledeća dva uslova:

$$f_k(x) \leq f_k(x^*), \quad \text{za} \quad k = 1, 2, \ldots, p \tag{1.5}$$

$$f_i(x) < f_i(x^*), \quad \text{za neko} \quad i \in \{1, 2, \ldots, k\} \tag{1.6}$$

gde p predstavlja broj kriterijuma, odnosno ciljeva, a X skup dopustivih rešenja. Rešenje x^* je slabo efikasno ako ne postoji x tako da važi:

$$f_k(x) < f_k(x^*), \quad \text{za svako} \quad k = 1, 2, \ldots, p \tag{1.7}$$

Nedominirani skup rešenja (non-dominated solution set) je skup svih rešenja koja nisu dominirana od strane bilo kog elementa skupa rešenja. Pareto-optimalni skup je nedominirani skup celog dopustivog prostora odlučivanja, a granica koju definišu tačke iz Pareto-optimalnog skupa sa naziva Pareto-optimalni front. Metode za rešavanje višekriterijumskih optimizacionih problema se mogu podeliti u skalarizacione (scalarization methods) i Pareto metode. U prvu grupu spadaju: pristup težinske sume (weighted sum approach), kompromisno programiranje (compromise programming), multiatributna utilitarna analiza (multiatribute utility analysis), ciljno programiranje (goal programming), leksikografski prisup, i fazi logika. Najpoznatije Pareto metode su: eksploracija i Pareto filterovanje, pristup težinske sume (sa težinskim skeniranjem), adaptivni metod težinske sume, normalni granični presek (normal boundary intersection) i višekriterijumske prirodne metaheuristike.

Algoritam se može definisati kao alat ili procedura za rešavanje nekog jasno određenog problema izračunavanja. On uzima skup vrednosti na ulazu nad kojima u nizu, odnosno, sekvenci koraka izračunavanja vrši obradu kako bi na kraju dao rezultat tj. izlaz. S obzirom da se u postavci problema specificira željeni odnos ulaznih i izlaznih vrednosti, zadatak algoritma je da odredi i opiše računarsku proceduru koja postiže taj željeni odnos. Algoritam mora biti tačan, što znači da mora da rešava definisani problem [7]. Vreme i prostor su dva najvažnija resursa koja algoritam ima na

raspolaganju. Vremenska složenost algoritma se definiše brojem koraka (u najgorem scenariju) neophodnih da bi se rešio problem veličine n. Prilikom određivanja vremenske složenosti algoritma mi ne saznajemo tačan broj koraka za koje se problem rešava, nego asimptotsku granicu, i tada koristimo veliku-O notaciju (Big-O notation).

Algoritmi čija je složenost manja ili jednaka polinomijalnoj se smatraju dovoljno dobrim ili pogodnim za primenu. Njihova složenost se obeležava sa $O(n^k)$, gde je k konstanta, a n veličina ulaza. Nažalost, postoji jedna grupa problema koji ne pripadaju gore pomenutoj kategoriji, ali je teško precizno utvrditi koji problemi iz te grupe se mogu rešiti u polinomijalnom vremenu, a koji ne. Razmotrimo sada probleme odlučivanja, odnosno, one probleme čiji je izlaz jedna binarna vrednost: DA ili NE. Mogu se definisati dve osnovne klase problema odlučivanja:

- P (polinomijalni) – problemi odlučivanja koji se mogu rešiti u polinomijalnom vremenu;
- NP (nedeterministički polinomijalni) – problemi sa sledećom karakteristikom: ako je odgovor DA, onda se provera ovoga može uraditi u polinomijalnom vremenu. Drugim rečima, ako imamo rešenje problema, možemo u polinomijalnom vremenu da uvrdimo da li je ono tačno.

Iako većina smatra da su klase problema P i NP različite, ovo pitanje još uvek nije našlo odgovor u formi preciznog dokaza. Ako postojanje polinomijalnog algoritma za neki problem X implicira postojanje polinomijalnog algoritma za svaki NP problem, onda se kaže da X pripada klasi NP-teških problema (NP-hard). Drugim rečima, NP-teški problemi su teški barem koliko i NP problemi. NP-kompletni problemi su oni problemi za koje važi sledeće: ako postoji polinomijalni algoritam za jedan NP-kompletan problem, onda sledi postojanje polinomijalnog algoritma za svaki NP-kompletan problem.

1.2 Metode rešavanja optimizacionih problema

Konvencionalni načini za rešavanje optimizacionih problema se mogu podeliti na: metode bazirane na matematičkoj analizi (calculus based methods), metode slučajnog pretraživanja (random-based search), ili nabrojive metode pretraživanja (enumerative search techniques). S druge strane, optimizacione metode se mogu klasifikovati i u

zavisnosti od toga da li su u optimizacionom procesu potrebni izvodi funkcije cilja ili ne, i tada govorimo o derivativnim i nederivativnim metodama. Derivativne metode, u koje npr. spadaju Njutnov metod, Gaus-Njutnov metod i sl., su bazirane na matematičkoj analizi, odnosno na gradijentnom pretraživanju. Ove klasične metode su dobre za rešavanje širokog spektra optimizacionih problema koji nemaju veliki broj promenljivih odlučivanja. Međutim, realni prolemi su obično diskretni i imaju veoma veliki broj promenljivih odlučivanja, odnosno, prostor pretraživanja je ogroman [8]. Za rešavanje ovakvih problema klasične metode ne daju zadovoljavajuće performanse i tada se pribegava korišćenju drugih, modernih metoda u koje spadaju heuristike, odnosno, metaheuristike.

Realni problemi sa kojima se susrećemo u procesu optimizacije su različitog nivoa složenosti, te je zbog toga veoma teško naći jedan metod ili algoritam pogodan za rešavanje svih vrsta problema. Teoreme „nema besplatnog ručka“ (no- free-lunch theorems, NFL) dokazuju da uopšte ne postoji univerzalni algoritam za sve probleme. NFL teoreme važe za determinističku ili stohastičku optimizaciju, gde je skup parametara kontinualan, diskretan ili mešovit, a skup vrednosti funkcije cilja je konačan. Osnovno tvrđenje ovih teorema je sledeće: Ako neki algoritam X pokazuje bolje performanse od drugog algoritma Y u potrazi za minimalnom ili maksimalnom vrednošću date funkcije cilja, onda će algoritam Y pokazati bolje performanse u slučaju neke druge funkcije cilja. Za dati problem, izbor pogodnog algoritma za rešavanje zavisi kako od iskustva korisnika, tako i od karakteristika algoritma kao što su: cena izračunavanja, dostupnost softvera, vreme izvršavanja itd. Ako za dati algoritam želimo da saznamo koje vrste problema može uspešno da rešava potrebno je da proverimo ponašanje algoritma pri rešavanju više problema različitih karakteristika, a onda da performanse tog algoritma uporedimo sa drugim algoritmima.

1.3 Test funkcije

Generalno gledano, evaluacija performansi nekog optimizacionog algoritma se može proveravati putem poznatih realnih ili veštačkih problema. Realni problemi mogu da pripadaju različitim fundamentalnim naučnim oblastima kao što su matematika, fizika, hemija, ili primenjenim discplinama poput inženjerstva, medicine i sl. Obično je matematički model ovakvih problema složen jer se opisuje komplikovanim mate-

matičkim izrazima i težak je za implementaciju. S druge strane, veštački problemi su test ili benčmark funkcije koje mogu da imaju različite karakteristike kao npr. jedan globalni minimum, više globalnih ili lokalnih minimuma, ravne površine itd [9]. Ovakvi problemi se mogu relativno lako modifikovati i prilagoditi za testiranje ponašanja optimizacionih algoritama u raznim situacijama. Do sada je razvijen i predložen veliki broj različitih test funkcija, mada ne postoji standardna lista benčmark funkcija za sve algoritme. Kada se želi uraditi ocena performansi nekog algoritma, mi u stvari hoćemo da utvrdimo za koje probleme taj algoritam pokazuje veću efikasnost u poređenju sa drugim algoritmima. Zbog toga skup funkcija koje u tom postupku koristimo ne smeju da budu suviše specijalizovane i međusobno sličnih karakteristika. Dakle, skup benčmark funkcija mora da sadrži različite funkcije, kojima se proverava efikasnost algoritama na različite probleme kao što su unimodalni, višemodalni, višedimenzionalni i sl. Samo ovakav pristup obezbeđuje da se jasno i precizno odredi za koje probleme je algoritam pogodan za primenu.

U praksi se često za ispitivanje performansi algoritama koriste skupovi funkcija koji su utvrđeni i primenjivani na naučnim skupovima i kongresima koji se bave oblastima optimizacije kao sto je IEEE CEC (Congress on Evolutionary Computation). Takvi skupovi funkcija se u literaturi obeležavaju oznakama koje sadrže skraćeni naziv kongresa i godinu održavanja (npr. CEC 2015).

1.4 Heuristike i metaheuristike

U mnogim realnim situacijama mi se suočavamo sa izuzetno komplikovanim problemima. Tada je upotreba optimizacionih metoda i tehnika neophodna, iako nam korišćenje ovih metoda ne garantuje da će optimalno rešenje uopšte biti nađeno. Da izazov bude veći, često takvi problemi pripadaju klasi NP-teških, za koje se ne mogu konstruisati efikasni algoritmi. Veliki broj ovakvih problema mora da se rešava pristupom „pokušaja i greške“ (trial and error) koristeći različite optimizacione tehnike [10]. U literaturi se može naći veliki broj NP-teških optimizacionih problema koji pripadaju različitim oblastima kao što su: teorija grafova, mrežni dizajn, matematičko programiranje, sekvenciranje i zakazivanje, logika, igre i zagonetke itd [11]. Da bi se pronašlo dovoljno dobro ili zadovoljavajuće rešenje za ovakve probleme, često se koriste metaheuristike. Generalno postoje dve vrste stohastičkih algoritama, heuri-

stike i metaheuristike. U literaturi ne postoji jasna definicija ova dva pojma, tako da se oni i ne mogu precizno razlikovati, te se vrlo često upotrebljavaju kao sinonimi. Termin heuristika je grčkog porekla i znači umetnost otkrivanja novih strategija za rešavanje problema. Obično se za heuristike smatra pretraga koja je bazirana na postupku pokušaja i greške (discover by trial and error). U najpoznatije heurističke metode spadaju planinarenje (hill climbing), metod grananja i ograničavanja (branch and bound), $A*$ i sl. Daljim razvojem heurističkih algoritama došlo je do pojave metaheuristika. Prefiks meta takođe potiče iz Grčke i znači „metodologija višeg nivoa“. Metaheurističke metode pretraživanja se mogu definisati kao opšte metodologije višeg nivoa koje se mogu koristiti kao strategije vodilje za dizajniranje heuristika kojima se rešavaju specifični optimizacioni problemi [12]. Drugim rečima, za razliku od heuristika koje se prave i adaptiraju u odnosu na specifični problem, metaheuristički algoritmi i metode su generalne strategije za rešavanje šireg opsega problema. Rešenje složenog optimizacionog problema dobijeno primenom metaheuristika je „dovoljno dobro“, i nalazi se u razumnom vremenu. Za razliku od tradicionalnih metoda, metaheuristički algoritmi ne garantuju nalaženje optimalnog rešenja. Takođe, za razliku od približnih optimizacionih metoda, metaheuristike ne daju informaciju o tome koliko je dobijeno rešenje blizu optimalnog.

U svim metaheuristikama se pravi kompromis između randomizacije i lokalne pretrage. U poslednje vreme čak postoji i tendencija da se svi stohastički algoritmi koji se baziraju na randomizaciji i lokalnoj pretrazi definišu kao metaheuristike. Najjednostavniji način za implementaciju stohasticiteta, odnosno randomizacije prilikom pretraživanja jeste slučajni hod (random walk). Obavlja se tako što se na slučajan način ide sa jednog mesta na drugo u prostoru pretraživanja kako bi se našlo optimalno rešenje. Takođe je važno istaći da je ogromna većina metaheuristika inpirisana prirodom, odnosno zasniva se na principima fizike, biologije ili etologije tj. ponašanja životinja.

Generalno posmatrano, metaheuristike se mogu podeliti na: individualne (single-solution based) i populacione (population based). Individualne metode se još nazivaju i metode trajektorije zato što algoritam prilikom izvršavanja pravi trajektoriju u prostoru pretraživanja, krećući se od jednog, inicijalnog rešenja ka drugim rešenjima koja se mogu nalaziti bilo gde u prostoru pretraživanja. Najpoznatiji algoritmi ovog tipa

su: simulirano kaljenje (simulated annealing, SA), tabu pretraživanje, GRASP metod, pretraživanje metodom promenljivih okolina (variable neighbourhood search, VNS), kao i razne varijante lokalnih pretraživanja [13]. Populacione metaheuristike rade sa više agenata, tj. populacijom individua, gde individue iz populacije međusobno dele informacije i na taj način zajednički tragaju za optimalnim rešenjem. U ovom slučaju se u prostoru pretraživanja prave višestruke trajektorije. Evolutivna izračunavanja (evolutionary computation, EC) i inteligencija rojeva (swarm intelligence, SI) su dve osnovne grupe koje pripadaju populacionim metaheuristikama. EC se bazira na Darvinovom procesu evolucije, gde se nad individuama primenjuje mutacija i rekombinacija, pri čemu one individue koje imaju najbolje karakteristike (koje definiše funkcija cilja) idu u sledeću generaciju. Inteligencija rojeva je skup metaheurističkih metoda optimizacije koje su inspirisane kolektivnim ponašanjem raznih vrsta životinja koje u međusobnoj saradnji i razmeni informacija ispoljavaju inteligenciju potrebnu za rešavanje složenih problema.

S obzirom da svi algortmi, uključujući i metaheuristike, imaju mane i nedostatke često se pribegava kombinovanju dva ili više komplementarnih algoritama kako bi se napravili novi hibridni algoritmi. Ovi algoritmi kombinuju prednosti svakog početnog algoritma ponaosob, dok istovremeno minimizuju nedostatke. Rezultat postupka hibridizacije je obično poboljšanje performansi u odnosu na početne algoritme, u smislu vremena izračunavanja ili tačnosti. Hibridni algorimi mogu biti kolaborativni i integrativni. Kolaborativni hibridi se sastoje od dva ili više algoritama koji rade sekvencijalno ili paralelno. Načini za implementaciju zajedničkog rada mogu biti: 1) više faza – između algoritama postoji podela poslova tako da npr. jedan algoritam radi globalnu optimizaciju, a drugi lokalno pretraživanje; 2) sekvencijalno – algorimi rade naizmenično, npr. oba algoritma rade određeni broj iteracija pre nego što izvršavanje prepuste drugom algoritmu; i 3) paralelno – algoritmi rade paralelno, na istoj populaciji. Kod integrativnih hibrida, jedan algoritam se smatra nadređenim, master algoritmom, dok je drugi podređeni (subordinate). Primer ovakve implementacije je inkorporacija određenih operatora iz jednog algoritma u drugi, primarni algoritam [14].

1.5 Prirodom inspirisane metaheuristike

U poslednjih nekoliko decenija, nova klasa optimizacionih algoritama inspirisana procesima koji se dešavaju u prirodi, dobija na značaju i popularnosti. Priroda ima dva ogromna i efikasna mehanizma: selekciju i mutaciju. Pomoću selekcije priroda nagrađuje individue koje se bolje prilagođavaju sredini u kojoj žive, tj. jače individue koje efikasnije rešavaju egzistencijalne probleme. Mutacija je mehanizam baziran na slučajnim procesima kojim se omogućuje rađanje novih, različitih individua. Upravo na ova dva mehanizma se zasniva prirodom inspirisana optimizacija i odgovarajući algoritmi. Naime, mehanizam selekcije je osnovni princip ovakve optimizacije, a mutacija je mehanizam koji stoji u osnovi stohastičkog pretraživanja. Postoje četiri osnovne osobine prirodom inspirisanih metaheuristika [15]:

- modeluju fenomen koji postoji u prirodi;
- imaju nedeterministički karakter;
- često implicitno imaju paralelnu strukturu (više agenata);
- imaju mogućnost adaptivnosti.

Idealni optimizacioni algoritam bi bio onaj koji u svakom sledećem ciklusu proizvodi bolja rešenja od onih u prethodnom. Takođe, bilo bi poželjno da se najbolje moguće rešenje dostigne nakon minimalnog broja iteracija. Međutim, takav algoritam još uvek nije razvijen, a verovatno nije ni moguć. Stohastički algoritmi imaju tu osobinu da nova rešenja ne moraju nužno da budu bolja od prethodnih. Iako je na prvi pogled nelogično, u toku rada ovakvih algoritama neophodno je povremeno, na slučajan način (randomizacijom) odabrati rešenja koja nisu najbolja, ali koja će nam pomoći da se proces pretraživanja ne zaglavi u lokalnom optimumu. Ovaj stohastički mehanizam, koji se naziva eksploracija ili diversifikacija, je u osnovi modernih metaheurističkih algoritama. On je globalnog karaktera i unosi diverzitet u proces pretraživanje tako što omogućava efikasnije pretraživanje udaljenih regiona u prostoru pretraživanja, tj. pretraživanje na globalnom nivou, i generisanje rešenja koja su udaljena i različita od postojećih. Na ovaj način eksploracija smanjuje verovatnoću da se algoritam zaglavi u lokalnom minimumu. Međutim, mana ovog procesa je spora kon-

vergencija jer se previše računarskih resursa troši na pretraživanje regiona koji mogu biti udaljeni od željenog globalnog optimuma [16].

Drugi važan mehanizam metaheuristika je eksploatacija ili intensifikacija. Ona podrazumeva fokus na lokalnu pretragu, odnosno pretraživanje u okolini postojećih rešenja, kako bi eksploatacijom informacija koje se od njih dobijaju generisala nova, bolja rešenja [17]. Informacije koje nam pomažu u ovom procesu su takođe lokalnog karaktera, poput gradijenta. Proces eksploatacije ubrzava konvergenciju algoritma, međutim prevelika eksploatacija može da zaglavi algoritam u lokalnom optimumu. Previše eksploracije, a premalo eksploatacije će povećati verovatnoću da se dostigne globalni optimum, međutim cena je spora konvergencija i trošenje resursa. S druge strane, previše eksploatacije, a premalo eksploracije će povećati brzinu konvergencije algoritma, ali će smanjiti verovatnoću da dobijeno rešenje bude globalno najbolje. S obzirom na osobine ova dva navedena procesa, njihove efekte i posledice, očigledno je da efikasan algoritam mora da obezbedi balans tj. kompromis između njih [18].

Prirodom inspirisane metaheuristike imaju osobine diverziteta i adaptacije. Adaptacija podrazumeva da je moguće promenom odgovarajućih parametara, kao što je veličina populacije ili drugi, algoritamski specifični parametri, postići bolje performanse. Što se tiče diverziteta, može se reći da je diverzitet direktna posledica randomizacije, i da se u zavisnosti od konkretnog algoritma može realizovati na različite načine, kao što je mutacija kod genetskih algoritama. Naravno, ove dve osobine su međusobno direktno povezane jer se, recimo, adaptacijom algoritma prilikom promene parametra može postići veći diverzitet [19].

U nastavku će biti opisano nekoliko najpoznatijih metaheuristika i to: 1) algoritam simuliranog kaljenja, kao tipični predstavnik metoda trajektorije ili individualnih metaheurističkih metoda; 2) genetski algoritam i diferencijalna evolucija – predstavnici populacionih metoda. O algoritmima inteligencije rojeva će biti više reči u sledećem poglavlju.

1.5.1 Simulirano kaljenje

Algoritam simuliranog kaljenja (simulated annealing, SA) je predložen početkom osamdesetih godina prošlog veka od strane Kirkpatrick-a i ostalih (1983), a zatim ga je nekoliko godina kasnije nezavisno od njih predložio i Černy (1985). Inspiracija za

stvaranje ovog algoritma je došla iz procesa kaljenja čvrstih materijala i analogije koja je uočena između ovog procesa i rešavanja problema kombinatorne optimizacije [20].

Iz fizike čvrstog stanja je poznato da se proces kaljenja sastoji iz dve faze. U prvoj fazi se čvrsti materijal zagreva do maksimalne vrednosti prelazeći u tečno stanje, kada se čestice u materijalu raspoređuju na proizvoljan i slučajan način. Nakon ove faze primenjuje se postepeno i sporo hlađenje materijala kada se dešava prelazak istog u čvrsto stanje kristalne rešetke sa najnižom energijom. U procesu hlađenja, sistem na svakoj temperaturi ulazi u termodinamičku ravnotežu koja je energetski određena Boltzmann-ovom raspodelom. Ukoliko je hlađenje dovoljno sporo, na temperaturi bliskoj nuli sistem ulazi u stanje minimalne energije. Međutim, ako se proces brzo odvija, odnosno ako se ne dozvoli da sistem na svakoj temperaturi ulazi u termodinamičku ravnotežu, događa se da materijal ostane u metastabilnom amorfnom stanju.

Za simuliranje evolucije materijala u stanje termodinamičke ravnoteže na datoj temperaturi predložen je Metropolis algoritam, nazvan po istraživaču koji ga je napravio 1953 godine. Algoritam se zasniva na Monte Carlo postupku koji generiše sekvencu stanja materijala na sledeći način. Pretpostavimo da se materijal nalazi u nekom stanju i koje je određeno položajem tj. rasporedom čestica i energijom sistema E_i. Promena stanja sistema i prelazak u stanje j sa energijom E_j se postiže primenom male, slučajno generisane perturbacije, kao što je izmeštanje čestice. Ako je razlika energija novog i tekućeg stanja, $\Delta E = E_j - E_i \leq 0$, novo stanje j se prihvata kao tekuće, u suprotnom, novo stanje se prihvata sa verovatnoćom $\exp(\frac{-\Delta E}{k_B T})$, gde je k_B Boltzmann-ova konstanta, a T temperatura. Ovo pravilo se naziva Metropolis kriterijum. Dakle, stanje termodinamičke ravnoteže se dostiže na iterativan način, generisanjem velikog broja tranzicija iz jednog stanja u drugo, po opisanoj proceduri.

Kombinatorni optimizacioni problem se može posmatrati kao proces kaljenja ako se uvedu sledeće dve analogije: 1) rešenja optimizacionog problema su ekvivalentna stanjima fizičkog sistema; 2) vrednost funkcije cilja je ekvivalentna energiji stanja. Za implementaciju algoritma neopohodan je i kontrolni parametar c koji menja temperaturu u izvornom Metropolis algoritmu. Na ovaj način, metoda simuliranog kaljenja se može posmatrati kao metoda pretraživanja realizovana kao iterativni niz Metropolis postupaka koji se sukcesivno izvršavaju tako da je u svakom sledećem izvršavanju vrednost kontrolnog parametra manja od vrednosti u tekućem izvršavanju. Algoritam

1.1 sadrži pseudokod metaheuristike simuliranog kaljenja [21].

Algoritam 1.1: Simulirano kaljenje

```
1  begin
2      Inicijalizacija (i_start, c_0, L_0);
3      k = 0;
4      i = i_start;
5      repeat
6          for i = 1 to L_k do
7              Generiši (j iz S_i);
8              if f_i ≤ f_j then
9                  i = j
10             else
11                 if exp (f_i − f_j)/c_k > random[0, 1) then
12                     i = j
13                 end
14             end
15         end
16         k = k + 1;
17         IzračunajDužinu(L_k);
18         IzračunajKtrlParam(c_k)
19     until kriterijum zaustavljanja;
20 end
```

Funkcija Inicijalizacija() na izlazu daje startno rešenje i radi inicijalizaciju kontrolnog parametra c i parametra L, koji predstavlja broj perturbacija u Metropolis algoritmu. Funkcija Generiši() služi za generisanje novih stanja (rešenja) iz susedstva postojećih, a funkcije IzračunajDužinu() i IzračunajKtrlParam() rade izračunavanje novih vrednosti parametara c i L. Brzina konvergencije algoritma zavisi od parametara c i L, a u konkretnim implementacijama algoritma promena parametra c se realizuje po formuli: $c_{k+1} = \alpha * c_k, (k = 0, 1, 2, \dots)$, gde se vrednosti konstante α uzimaju iz intervala 0.8 do 0.99. SA algoritam je našao široku primenu u rešavanju jednociljnih i višeciljnih problema kombinatorne optimizacije. Dodatno, razvijen i veliki broj hibridnih algoritama u kojima je jedna komponenta SA algoritam [22].

1.5.2 Genetski algoritmi

Procesi evolucije i prirodne selekcije su bili inspiracija za razvoj genetskih algoritama (genetic algorithms, GA). Ovi algoritmi spadaju u populacione, evolutivne metaheuristike. Genetski algoritmi su stekli popularnost na osnovu dve komparativne prednosti u odnosu na ostale algoritme: 1) sposobnost rešavanja složenih optimizacionih problema različitih tipova i sa različitim funkcijama cilja; 2) paralelizam, koji je moguć zahvaljujući činjenici da su višestruki geni pogodni za paralelnu optimizaciju. Osnovne komponente GA algoritama su tzv. genetski operatori nasleđivanja, mutacije, ukrštanja i selekcije. Na početku rada algoritma inicijalizuje se populacija individua koja se zatim razvija primenom genetskih operatora. Individue tj. rešenja su predstavljena u obliku binarnog ili decimalnog stringa koji se naziva hromozom, dok su geni pojedinačni biti ili nizovi susednih bita koji kodiraju određeni element rešenja [23].

Dodavanje novih individua se obavlja operatorima ukrštanja i mutacije. Primenom operatora ukrštanja roditeljskih individua se dodaju nove individue u populaciju, pri čemu je verovatnoća ukrštanja velika, od 0.7 do 1.0. Operacija se realizuje tako što algoritam na slučajan način odabere poziciju unutar hromozoma i izvrši razmenu odgovorajućih sekvenci bita roditelja pre i posle ove pozicije kako bi se kreirale dve individue. Realizacija koncepta preživljavanja najjačih i najboljih u mehanizmu prirodne selekcije se u GA algoritmima obavlja preko operatora selekcije. U najjednostavnijoj verziji GA algoritma selekcija se obavlja po proporcionalnoj šemi što znači da se individui sa fitnes vrednošću f alocira f/f_{sr} potomaka [24]. Rešenje koje ima fitnes vrednost veću od f/f_{sr} dobija više od jednog potomka, a ono rešenje sa fitnes vrednošću manjom od f/f_{sr} dobija manje od jednog potomka. S obzirom da je f/f_{sr} očekivani broj potomaka, u krajnjem odlučivanju o broju potomaka se uvodi randomizacija kako bi se smanjila pristrasnost prema određenim individuama. Najjednostavniji način implementacije proporcionalne šeme je preko selekcije ruletom (roulette wheel selection). U ovom mehanizmu se svakoj individui dodeljuje deo (sektor) točka ruleta proporcionalan vrednosti f/f_{sr}. Određenoj individui se alocira potomak ako na slučajan način generisana vrednost u opsegu 0 do 2π pripada sektoru koji joj je dodeljen. Osim ovog mehanizma, najčešće se koristi još i turnirska selekcija (tournament selection). Dodatno se koristi i elitistička strategija koja omogućuje da individue sa najboljom

vrednošću fitnes funkcije budu kopirane u sledeću generaciju [25]. Operacija mutacije se realizuje inverzijom nekih bita u hromozom stringu, i ima manju verovatnoću, obično od 0.001 do 0.05. Rešenja se evaluiraju u skladu sa funkcijom cilja optimizacionog problema, a procedura selekcije bira samo najbolja rešenja za sledeću generaciju [26]. Pseudokod osnovnog GA algoritma je prikazan u Algoritmu 1.2.

Algoritam 1.2: Genetski algoritam

```
1  begin
2      Definiši fitnes funkciju F;
3      InicijalizujPopulaciju();
4      Inicijalizuj verovatnoće ukrštanja p_c i mutacije p_m;
5      EvaluirajPopulaciju();
6      while nije dostignut kriterijum zaustavljanja do
7          Izvrši selekciju individua za sledeću generaciju;
8          Obavi ukrštanje sa verovatnoćom p_c;
9          Obavi mutaciju sa verovatnoćom p_m;
10         EvaluirajPopulaciju();
11     end
12 end
```

Kao i kod ostalih metaheuristika, izbor kontrolnih parametara je važno pitanje u GA algoritmima. Ako je verovatnoća ukrštanja prevelika, može da dođe do preuranjene konvergencije i zaglavljivanja u lokalnom optimumu. S druge strane, ako je ova verovatnoća veoma mala, ukštanje se ređe dešava što sprečava evoluciju i čini ceo algoritam manje efikasnim. Mutacija je operator kojim se suštinski ostvaruje eksploracija i obezbeđuje diverzitet rešenja. Mala verovatnoća mutacije može da spreči pojavu generisanja rešenja koja su različita od postojećih, a velike vrednosti da izazovu preveliko „lutanje“ algoritma u prostoru pretraživanja i da uspore kovergenciju, čak i kada je rešenje relativno blizu. Veličina populacija ne sme biti previše mala da se ne bi ugrozili evolucija i diverzitet, i došlo do preuranjene konvergencije. Međutim, prevelika populacija rezultuje u povećanju troškova izračunavanja. Najbolje vrednosti za veličinu populacije su između 40 i 200, zavisno od vrste optimizacionog problema.

1.5.3 Diferencijalna evolucija

Diferencijalna evolucija (Differential Evolution, DE) je paralelni direktni metod pretraživanja koji pripada grupi evolutivnih algoritama. Populacija bilo koje generacije G je u ovom algoritmu predstavljena pomoću N D-dimenzionalnih vektora: $x_i, i = 1, 2, \ldots N$. U fazi inicijalizacije komponentama vektora se dodeljuju slučajne vrednosti iz uniformne raspodele, a broj vektora N se ne menja u toku rada algoritma. U slučaju da je poznato preliminarno rešenje, inicijalna populacija može biti generisana dodavanjem slučajnih vrednosti iz normalne raspedele nominalnom rešenju $x_{nom,0}$ [27]. Kako što je slučaj i kod ostalih evolutivnih algoritama, DE ima tri glavna operatora: mutaciju, rekombinaciju (ukrštanje) i selekciju. Operacija mutacije se izvršava za svaki vektor x_i, tako što se na slučajan način biraju tri različita vektora x_t,x_r i x_s, i pravi mutirajući vektor x_m:

$$x_m = x_t + F * (x_r - x_s) \tag{1.8}$$

gde je $F > 0$ težinski ili skalirajući faktor koji pojačava diferencijalnu varijaciju dva vektora. Autori algoritma su predložii da vrednost F bude u intervalu $[0, 2]$, a praksa je pokazala da su najefektivnije vrednosti retko veće od 1. Operacija mutacije osim jednačinom (1.8) može biti realizovana i na druge načine [28]. Generisanje nove individue se obavlja nakon operacije ukrštanja, na sledeći način:

$$x_{new,j} = \begin{cases} x_{i,j}, & \text{ako je} \quad rand(0,1) < CR \\ x_{m,j}, & \text{inače} \end{cases} \tag{1.9}$$

gde je $rand(0, 1)$ slučajna promenljiva iz uniformne raspodele, a CR konstanta ukrštanja (crossover constant), koja se uzima iz intervala [0,1] i njenu vrednost definiše korisnik. Za svaku komponentu j vektora nove individue x_{new} se na probabilistički način definiše da li će uzeti vrednost početnog vektora x_i ili mutirajućeg vektora x_m, čime se suštinski realizuje ukrštanje između njih. Pseudokod osnovnog DE algoritma je dat u Algoritmu 1.3 [29].

S obzirom da su u početnim iteracija DE algoritma rešenja široko raspodeljena u prostoru pretraživanja, veličina koraka $F(x_r - x_s)$ ima veće vrednosti te će dominirati proces eksploracije. U kasnijim fazama algoritma se rešenja koncentrišu u određenim

regionima prostora pretraživanja tako da se veličina koraka postepeno smanjuje. Na ovaj način pretraživanje postaje lokalizovano, odnosno, dominira proces eksploatacije. Iako je ovakvo ponašanje algoritma poželjno, ono u sebi nosi i određeno ograničenje i potencijalnu manu. Naime, ako algoritam ne uspe u generisanju novih individua koje su bolje od tekućih, veličina koraka se neće smanjivati i doći će do stagnacije koja će sprečiti konvergenciju algoritma ka suboptimalnim rešenjima.

Algoritam 1.3: Diferencijalna evolucija

```
1  begin
2      Na pseudoslučajan način generiši N individua inicijalne populacije;
3      while kriterijum zaustavljanja do
4          for i = 1 to N do
5              Izračunaj f(x_i)
6          end
7          for i = 1 to N do
8              Izabrati tri individualna vektora x_r, x_s, i x_t;
9              Izračunaj x_m po jednačini (1.8);
10             x_new = x_m;
11             for j = 1 to D do
12                 Generiši rand(0, 1);
13                 if rand(0, 1) < CR then
14                     x_new,j = x_i,j
15                 end
16             end
17             if f(x_new) ≤ f(x_i) then
18                 Sačuvaj indeks radi zamene x_i = x_new
19             end
20         end
21         Izvrši zamenu;
22     end
23 end
```

Očigledno je da su performanse algoritma veoma zavisne od izbora kontrolnih parametara, u koje spadaju veličina populacije N, težinski faktor F i konstanta ukrštanja CR. Pokazalo se da je izbor vrednosti parametara F i CR izuzetno težak, i tokom godina istraživači su predložili veliki broj rešenja za podešavanje njihovih vrednosti tako da efikasnost DE algoritma bude maksimalno moguća [30].

1.6 Primena optimizacije

Kao što je već rečeno na početku ovog poglavlja, optimizacija se primenjuje u mnogim oblastima i disciplinama. Ovde ćemo navesti jedan od primera primene, kojim smo se bavili u toku dosadašnjeg naučnoistraživačkog rada. Naime, predmet optimizacije može da bude i redudancija nanotehnoloških elektronskih komponenti. Stalno smanjenje dimenzija ovih komponenti uz povećanu elektromagnetnu kontaminaciju životne sredine i uvek prisutno sekundarno kosmičko zračenje, znatno smanjuju pouzdanost rada takvih komponenti. Ovo posebno dolazi do izražaja u izradi nanotehnoloških računara. Rešenje za ovaj problem je redudancija memorijskih i ostalih sistema računara. Međutim, ovakvo rešenje donosi povećanje troškova potrošnje energije, te je nužno optimizovati stepen redudancije, uz postizanje maksimalne pouzdanosti rada. Tako je u [31] razmatran problem optimizacije postupka redudancije MOS memorijskih struktura nanotehnološki izrađenih računara koji rade u radijacionom okruženju.

Na osnovu rezultata eksperimenata se zaključuje da bi jedna MOS struktura izložena dejstvu pozadinskog zračenja imala verovatnoću od 99% da sadrži pogrešan zapis. Gledajući tako, nano računari bi bili nemogući za primenu u realnim uslovima. Usled toga je napuštena ideja izrade nano računara sa pojedinačnim integrisanim komponentama, već se konstruišu nano strukture u kojima su komponente redundovane (redno i serijski). Naravno, iako su nano komponente veoma jeftine, broj redundovanih se mora minimizovati zbog potrošnje struje na rad takvog računara kao i na njegovo hlađenje.

U slučaju paralelne veze diskretnih, nezavisnih MOS memorija izrađenih u nanotehnološkoj tehnologiji, faktor uvećanja je jednodimenzionalan i jednak broju korišćenih memorija. Posmatrajući navedeni primer sa faktorom uvećanja n=10000 i verovatnoćom da pojedinačni nanotehnološki izrađen MOS memorijski sklop ima pogrešan zapis izazvanog pozadinskim zračenjem od 0.005, onda je verovatnoća da se taj pogrešan sadržaj pojavi u celom, redundovanom memorijskom sklopu biti $0.05 * 10000^{-1}$, odnosno ako se pretpostavi da 100 pojedinačnih nanotehnoloških MOS struktura ima istu verovatnoću imanja pogrešnog sadržaja, informacija iz redundovanog sklopa je 0,000005 statistički pouzdana. U standardnim uslovima ovo je zadovoljavajuća statistička pouzdanost zaključivanja.

2 INTELIGENCIJA ROJEVA

2.1 Uvod

Inteligencija rojeva (swarm intelligence) je prirodom inspirisana paradigma za rešavanje optimizacionih problema. Ideja za njenu implementaciju je dobijena posmatranjem ponašanja bioloških sistema, tačnije, socijalnih insekata kao što su mravi, pčele, svici, i drugih životinja kao što su jata ptica ili riba. Osim ovoga, inspiracija za razvoj ove vrste algoritama je dolazila i iz drugih fenomena u prirodi, kao što je eksplozija vatrometa, ili iz socijalnih fenomena, poput ponašanja ljudi prilikom generisanja ideja i rešavanja problema. Osnovne karakteristike ove vrste algoritama su samoorganizacija, koevolucija, distribuiranost i decentralizovan pristup.

Mnogi biološki organizmi u grupi se ponašaju tako da donose odluke na osnovu lokalnih informacija koje dobijaju od spoljašnje sredine, kao i putem interakcija sa ostalim organizmima iz grupe. Ove interakcije na kraju mogu da dovedu do pojave kolektivne ili socijalne inteligencije. Pretpostavka je da se ovo dešava zato što su biološke promene kod organizama posledica njihovih adaptacija na promene u okruženju i u grupi kojoj pripadaju, te je pojava inteligencije kod razvijenih bioloških vrsta direktna ili indirektna posledica složenih interakcija. Istraživanja su pokazala da grupe organizama iste vrste imaju sposobnost rešavanja kompleksnih zadataka pomoću kolektivne inteligencije tj. inteligencije rojeva [32]. Polazeći od fenomena inteligencije rojeva, naučnici su razvili veliki broj metoda i algoritama za rešavanje složenih problema, u koje spadaju optimizacije na bazi ponašanja insekata ili drugih životinja. Primena ovih algoritama ima smisla zato što iterativni proces algoritma liči na samoorganizujuću evoluciju sistema [33]. S obzirom da je slučajnost inherentna osobina ovih algoritama, to povećava verovatnoću da rešenje neće biti zaglavljeno u lokalnom minimumu, odnosno, da će algoritam brže da konvergira ka globalnom optimalnom rešenju. Samoorganizacija u rojevima ima sledeće četiri karakteristike [34]:

1. Pozitivne povratne informacije: služe promociji i podsticanju na kreiranje podesnih struktura. Feromonski tragovi koje ostavljaju mravi su primer pozitivnih povratnih informacija;

2. Negativne povratne informacije: protivteža pozitivnim povratnim informacijma

i služe za stabilizaciju populacije. Ovaj mehanizam je potreban kako ne bi došlo do zasićenja npr. u broju dostupnih tragalaca za hranom (foragers);

3. Fluktuacije: slučajnost u kretanju, obavljanju zadataka i sl. su od vitalnog značaja za očuvanje kreativnosti jer omogućuju otkrivanje novih rešenja;

4. Višestruke interakcije: svaka individua koristi, šalje i prima informacije od ostalih članova roja (jata), i na taj način se informacije šire kroz roj.

U osnovi, sistem inteligencije rojeva se bazira na veoma prostim pravilima. Individualni agenti, npr. mravi ili pčele, donose odluke na osnovu lokalnih informacija, pri čemu ne postoji nikakva centralizovana kontrola ili entitet koji utiču na to kako agenti treba da se ponašaju. Ponašanje agenata je lokalno, u određenoj meri slučajno, ali interakcije među agentima dovode do pojave samoorganizacije i globalno „inteligentnog“ ponašanja koje nije poznato individualnim agentima. Uslovi koji su neophodni da bi došlo do samoorganizacije unutar sistema su: povratne informacije, stigmergija (kada individue komuniciraju indirektno putem modifikacija lokalnog okruženja), višestruke interakcije, memorija, i uslovi u okruženju [34].

U algortmima inteligencije rojeva, svaka individua predstavlja rešenje u prostoru pretraživanja. Ovi algoritmi moraju da imaju dve vrste mogućnosti: učenje o sposobnostima i razvoj kapaciteta [35]. Razvoj kapaciteta se odnosi na kretanje algoritamske pretrage prema oblastima gde postoji veći potencijal za pretraživanje, dok se učenje o sposobnostima fokusira na samo pretraživanje, i to polazeći od trenutnog rešenja, za optimizacione algoritme sa jednom tačkom, odnosno, od trenutne populacije, za populaciono bazirane algoritme inteligencije rojeva. Algoritmi inteligencije rojeva koji imaju obe mogućnosti se nazivaju razvojni algoritmi inteligencije rojeva (developmental swarm intelligence algorithms, DSI).

Optimizacija kolonijom mrava (ant colony optimization, 1992), optimizacija rojevima čestica (particle swarm optimization, 1995), bakteriološki algoritam (bacterial foraging, 2002), algoritam veštačke kolonije pčela (artificial bee colony, 2005), majmunsko pretraživanje (monkey search, 2007), algoritam svica (firefly algorithm, 2008), kukavičje pretraživanje (cuckoo search, 2009), algoritam sjajnog crva (glowworm algorithm, 2009), algoritam slepog miša (bat algorithm, 2010), algoritam vatrometa (fireworks algorithm, 2010), brain storm algoritam (2011), algoritam kril jata (krill

herd algorithm, 2012), algoritam sivog vuka (gray wolf optimization, 2014), lavlji algoritam (lion optimization algorithm, 2016) su neki od najpoznatijih algoritama koji pripadaju inteligenciji rojeva. U nastavku ovog poglavlja biće ukratko opisano nekoliko najznačajnijih algoritama inteligencije rojeva, a brain storm algoritam će biti detaljno opisan u narednom poglavlju.

2.2 Algoritam kolonije mrava

Algoritmi mrava su uvedeni kao pristup za rešavanje multiagentnih kombinatornih optimizacionih problema kao što je problem putujućeg trgovca ili problem kvadratne asignacije. Inspiracija za razvoj ovih algoritama je dobijena posmatranjem ponašanja kolonije mrava, tačnije, kako mravi nalaze najkraći put od njihovog gnezda do izvora hrane [36]. Analogno biološkim mravima, algoritam kolonije mrava (ant colony optimization, ACO) se zasniva na indirektnoj komunikaciji kolonije jednostavnih agenata, koji se nazivaju veštačkim mravima, putem veštačkih tragova feromona (pheromone trails). Ovi tragovi služe kao distribuirane numeričke informacije koje mravi koriste kako bi na probabilistički način konstruisali rešenje problema i koje mravi adaptiraju i ažuriraju tokom rada algoritma da bi bolje odslikavali njihovo iskustvo pretraživanja [37]. ACO algoritmi koriste populaciju mrava da bi kolektivno rešili neki problem. Informacije koje mravi prikupljaju u toku procesa pretraživanja se čuvaju u tragovima feromona koji su pridruženi odgovarajućim vezama, i ovi tragovi predstavljaju dugotrajnu memoriju o datom procesu pretraživanja. Kolonija mrava ima sledeće osobine [38]:

- Mrav traži rešenje koje ima najmanje troškove;
- Mrav k ima memoriju M_k, koju koristi kako bi sačuvao informaciju o putanji koju je prešao do tekućeg trenutka. Memorija se koristi da bi se generisala dopustiva rešenja, za evaluaciju nađenih rešenja, kao i za restauraciju putanje unazad;
- Mrav k u stanju s može da ide ka bilo kojem čvoru j u svom pogodnom okruženju;
- Mravu se može dodeliti startno stanje i jedan ili više uslova završetka;
- Mravi polaze od startnog stanja i kreću se ka pogodnim susednim stanjima, gradeći rešenje na inkrementalan način. Procedura se prekida kada je makar za

jednog mrava ispunjen barem jedan uslov završetka;

- Mrav k lociran u čvoru i može da se kreće ka izabranom čvoru j iz svog pogodnog okruženja, pri čemu je kretanje definisano pravilom o probabilističkom odlučivanju (probability decision rule);

- Pravilo o probabilističkom odlučivanju je funkcija: (1) vrednosti koje se čuvaju u lokalnoj strukturi podataka čvora koja se naziva tabela rutiranja mrava (ant routing table), a koja se dobija funkcionalnom kompozicijom tragova feromona koji su lokalno dostupni čvoru i heurističkih vrednosti, (2) privatne memorije mrava koja čuva istoriju kretanja mrava, i (3) definisanih ograničenja problema;

- Kada se kreće iz čvora i ka susednom čvoru j, mrav može da ažurira trag feromona na toj vezi. Ovo se naziva korak-po-korak ažuriranje feromona;

- Kada se kreira rešenje, mrav može da rekonstruiše putanju unazad i da ažuirira tragove feromona na vezama. Ovo se naziva odloženo ažuriranje feromona;

- Nakon toga, mrav umire i oslobađa alocirane resurse.

Centralna komponenta ACO algoritma je parametrizovani probabilistički model koji se naziva feromonski model. Ovaj model se sastoji od vektora model parametara nazvanih parametri tragova feromona, koji su obično povezani sa komponentama rešenja i imaju vrednosti koje se zovu feromonske vrednosti. Feromonski model se koristi za probabilističko generisanje rešenja od konačnog skupa komponenti rešenja. Tokom izvršavanja algoritma, ACO ažurira vrednosti feromona koristeći prethodno generisana rešenja. Cilj ažuriranja je da se pretraživanje fokusira na regione koji sadrže kvalitetna rešenja [39]. Uopštena struktura ACO optimizacionog algoritma je data u Algoritmu 2.1.

Nakon inicijalizacije parametara i feromonskih tragova, glavni deo algoritma se sastoji od tri koraka, koja se ponavljaju u svakoj iteraciji algoritma dok se ne dostigne neki uslov prestanka rada. Prvo se radi konstrukcija rešenja u skladu sa informacijama od feromona i, eventualno, dostupnim heurističkim informacijama. Kada mravi kompletiraju konstrukciju rešenja, opciono se ona mogu poboljšati u koraku daemon akcije, odnosno lokalne pretrage. Na kraju se radi ažuriranje vrednosti feromona kako

Algoritam 2.1: ACO algoritam

```
1 begin
2     Inicijalizacija;
3     while nisu ispunjeni kriterijumi završetka do
4         KonstrukcijaRešenja;
5         DaemonAkcije (opciono);
6         AžuriranjeFeromona;
7     end
8 end
```

bi bile usklađene sa iskustvom pretraživanja [40]. Konstrukcija rešenja se odvija inkrementalno, tako što svaki mrav počinje sa praznim rešenjem (s_p je prazan skup), a onda u toku rada algoritma svoje parcijalno rešenje nadograđuje i proširuje dodavanjem jedne pogodne komponente (c_i^j) iz skupa komponenti $N(s_p)$. Dati skup je implicitno određen procesom konstrukcije rešenja koji mrav implementira. Odluka koja komponenta iz $N(s_p)$ će se odabrati se donosi u svakom koraku konstrukcije rešenja na probabilističi način, primenom tzv. tranzicionih verovatnoća, a u skladu sa feromonskim modelom. Postoje različiti načini za definisanje raspodela verovatnoća koje se pritom koriste, ali u većini ACO algoritama se koristi pravilo Mravljeg sistema (Ant System) koje je definisano sledećom jednačinom [41]:

$$p(c_i^j \mid s_p) = \frac{\tau_{i,j}^{\alpha}[\eta(c_i^j)]^{\beta}}{\sum_{c_i^l \in N(s_p)} \tau_{i,l}^{\alpha}[\eta(c_i^l)]^{\beta}} \tag{2.1}$$

gde je: $\eta()$ funkcija koja svakoj dopustivoj komponenti rešenja c_i^j dodeljuje heurističku vrednost (heurističke informacije); α i β određuju uticaj tragova feromona i heurističkih informacija. Različite verzije ACO algoritma se razlikuju po tome kako vrše ažuriranje feronomskih vrednosti koje primenjuju. Cilj ažuriranja feromona je da komponente koje pripadaju dobrim rešenjima učine poželjnijim za selektovanje od strane drugih mrava u narednim iteracijama. Mehanizam ažuriranja feromona se sastoji iz dva dela. Prvo se radi operacija isparavanja feromona (pheromone evaporation), što predstavlja uniformno smanjivanje feromonskih vrednosti koje su postavili mravi u prethodnim iteracijama, sa ciljem da se izbegne veoma brza konvergencija algoritma

ka suboptimalnim regionima. Ova operacija primenjuje koristan oblik zaboravljanja (forgetting), što favorizuje nove oblasti u prostoru pretraživanja. Druga operacija je feromonski deposit (pheromone deposit) kojim se uvećava nivo feromona onih komponenti rešenja koje pripadaju skupu dobrih rešenja, S_{upd}. Ažuriranje feromona se obično obavlja u skladu sa sledećom jednačinom [42]:

$$\tau_{ij} = (1 - \rho) + \sum_{s \in S_{upd} | c_i^j \in s} g(s) \tag{2.2}$$

gde je ρ – brzina isparavanja (evaporation rate), a $g(s)$ – evaluaciona funkcija.

Daemon akcije su centralizovane procedure koje mravi samostalno ne mogu da izvršavaju. Primer ovakve akcije je primena dodatnog feromonskog depozita na komponente rešenja koje su se pokazale najboljim. Takođe, konstruisana kandidatska rešenja se mogu dodatno poboljšati primenom algoritama lokalnog pretraživanja kao daemon akcije.

2.3 Optimizacija rojevima čestica

Algoritam za optimizaciju rojevima čestica (particle swarm optimization algorithm, PSO) je stohastički metod za optimizaciju razvijen od strane Eberhart-a i Kennedy-ja 1995. godine. Zasniva se na simulaciji socijalnog ponašanja jata životinja kao što su ribe ili ptice, koje prilikom potrage za hranom primenjuju princip kooperacije, tako da svaki član jata kontinuirano menja svoj način potrage u skladu sa informacijama koje dobija od okruženja. Da bismo bolje razumeli pojavu i razvoj algoritma optimizacije rojevima čestica, upoznaćemo se sa jednostavnim Boid modelom [43] koji je uveden da bi simulirao ponašanje ptica i koji je poslužio kao osnova algoritma optimizacije rojevima čestica. U ovom modelu svaka ptica se predstavlja tačkom u Dekartovom koordinatnom sistemu, kojoj se dodeljuju inicijalna pozicija i brzina. S obzirom da se simulacija odvija u skladu sa pravilom koje kaže da indivudua teži da ima istu brzinu kao najbliži susedi, vrlo brzo će sve tačke imati istu brzinu, što je veoma jednostavan model i ne odgovara realnom scenariju. Da bi se ovo sprečilo, tj. da bi simulacija bliže odslikavala realnu situaciju, brzinama se u svakoj iteraciji dodaje slučajna vrednost.

U Heppner-ovom modelu kukuruza [44] pretpostavlja se da su na početku simu-

lacije položaj hrane, kao i položaj i brzina ptica slučajno određeni. Koordinate polja kukuruza su (x_0, y_0), dok su koordinate položaja i brzine ptica (x, y) i (v_x, v_y), respektivno. Performanse trenutnog položaja i brzine ptice se određuju na osnovu udaljenosti ptice i hrane; što je udaljenost veća performanse su bolje, i obrnuto. Ako pretpostavimo da ptice imaju sposobnost pamćenja najboljeg stanja i usvojimo sledeću notaciju: *pbest* – najbolja pozicija, a – konstanta za podešavanje brzine, *rand* – slučajan broj u intervalu $[0, 1]$, onda važi:

$$v_x = \begin{cases} v_x - rand * a, & \text{ako je} \quad x > pbestx \\ v_x + rand * a, & \text{u suprotnom} \end{cases} \tag{2.3}$$

$$v_y = \begin{cases} v_y - rand * a, & \text{ako je} \quad y > pbesty \\ v_y + rand * a, & \text{u suprotnom} \end{cases} \tag{2.4}$$

Ako takođe pretpostavimo da ptice u jatu mogu međusobno da komuniciraju i da zapamte najbolju poziciju jata (*gbest*) onda važi i sledeće:

$$v_x = \begin{cases} v_x - rand * b, & \text{ako je} \quad x > gbestx \\ v_x + rand * b, & \text{u suprotnom} \end{cases} \tag{2.5}$$

$$v_y = \begin{cases} v_y - rand * b, & \text{ako je} \quad y > gbesty \\ v_y + rand * b, & \text{u suprotnom} \end{cases} \tag{2.6}$$

Polazeći od gore navedenih modela, Kennedy i Eberhart su napravili novi optimizacioni algoritam kojeg su nazvali algoritam optimizacije rojevima čestica, zato što se u njihovom rešenju pojedinačne ptice posmatraju kao čestice bez mase i zapremine, gde su od interesa samo njihova brzina i položaj. Pravilo po kojem se ažuriraju brzina i položaj čestice u ovom algoritmu se može predstaviti sledećim izrazima [45]:

$$v_x = v_x + 2 * rand * (pbestx - x) + 2 * rand * (gbestx - x) \tag{2.7}$$

$$x = x + v_x \tag{2.8}$$

U algoritmu optimizacije rojevima čestica, svaka čestica predstavlja potencijalno

rešenje optimizacionog problema u D-dimenzionalnom prostoru pretraživanja. S obzirom da svaka čestica pamti svoju najbolju poziciju, kao i najbolju poziciju jata, u svakoj iteraciji na osnovu datih informacija čestice mogu da izračunaju novu brzinu, i na taj način da odrede novu poziciju. PSO algoritam radi tako što u svakoj iteraciji čestice podešavaju svoju brzinu tako da se kreću u pravcu prethodno najbolje pozicije (*pbest*) i globalno najbolje pozicije u jatu (*gbest*), pri čemu je [46]:

$$\forall i \in \{1, 2, \ldots, N\} \; pbest(t) = arg \; min[f(P_i(k))], k = 1, 2, \ldots, t \tag{2.9}$$

$$gbest(t) = arg \; min[f(P_i(k))], i = 1, 2, \ldots, N, k = 1, 2, \ldots, t \tag{2.10}$$

gde je: i – indeks čestice, N – ukupan broj čestica, t – trenutna iteracija, f – fitnes funkcija, a P – pozicija čestice. Uopštene formule za ažuriranje brzine i položaja čestice su [47]:

$$V_i(t+1) = \omega V_i(t) + c_1 r_1(pbest(i,t) - P_i(t)) + c_2 r_2(gbest(t) - P_i(t)) \tag{2.11}$$

$$P_i(t+1) = P_i(t) + V_i(t+1) \tag{2.12}$$

gde je V – brzina čestice, ω – težinski faktor inercije, koji se koristi radi uravnoteženja globalne eksploracije i lokalne eksploatacije, r_1 i r_2 su slučajne promenljive iz uniformne raspodele iz opsega $[0, 1]$, a c_1 i c_2 su pozitivne konstante koje se nazivaju koeficijenti ubrzanja. U Algoritmu 2.2 je dat pseudokod standardnog PSO algoritma [47].

U cilju poboljšanja originalnog PSO algoritma izvršen je veliki broj istraživanja i predložen je ogroman broj različitih varijanti. Izmene PSO algoritma mogu da se odnose na: strukturu algoritma, promenu parametara ili promenu topologije. Promena strukture algoritma može da se implementira na više načina od kojih su najpoznatiji sledeći: uvođenje podpopulacija, izmene strategije ažuriranja brzine ili položaja čestice, primena tehnika za očuvanje diverziteta populacije ili kombinacijama sa drugim prirodom inspirisanim algoritmima kako bi se kreirali hibridni algoritmi. Takođe, izmene u strukturi su neophodne i prilikom rešavanja višeciljnih i multimodalnih problema, kao i problema koji uključuju diskretne ili binarne promenljive.

Algoritam 2.2: Standardni PSO algoritam

begin
  Za svaku česticu $i = 1, 2, \ldots, N$:
  Inicijalizovati pozicije čestica sa uniformnom raspodelom iz intervala $[LB, UB]$, $(0) \sim U(LB, UB)$, gde LB i UB predstavljaju donju i gornju granicu prostora pretraživanja;
  Inicijalizovati *pbest* kao $pbest(i, 0) = P_i(0)$;
  Inicijalizovati *gbest* na minimalnu vrednost jata: $gbest(0) = arg\ minf[P_i(0)]$;
  Inicijalizovati brzine: $V_i \sim (-|UB - LB|, |UB - LB|)$;
  while *nije ispunjen kriterijum prekida rada* **do**
    Za svaku česticu $i = 1, 2, \ldots, N$:
    Izabrati slučajne vrednosti r_1, r_2, koje pripadaju uniformnoj raspodeli,r_1, r_2, $\sim U(0, 1)$;
    Modifikovati brzine čestica po formuli (2.7);
    Modifikovati položaj čestica po formuli (2.8);
    if $f[P_i(t)] < f[pbest(i,t)]$ **then**
      Ažurirati najbolju poziciju čestice i: $pbest(i,t) = P_i(t)$;
      if $f[P_i(t)] < f[gbest(t)]$ **then**
        Ažurirati najbolju poziciju u jatu: $gbest(t) = P_i(t)$
      end
    end
    $t \leftarrow (t + 1)$;
  end
  Izlazna vrednost $gbest(t)$ sadrži najbolje rešenje;
end

2.4 Algoritam veštačke kolonije pčela

Model pčela u potrazi za hranom ima sledeće tri komponente: izvori hrane, zaposleni tragaoci i nazaposleni tragaoci. Ciljevi kolonije pčela su zajedničko angažovanje u pronalasku dobrih izvora hrane i napuštanju nekvalitetnih izvora hrane. Izvori hrane se procenjuju na osnovu više osobina kao što su udaljenost od košnice, lokacija, bogatstvo nektarom, i lakoća izvlačenja hrane. Zaposleni tragaoci (employed foragers) su oni koji su trenutno angažovani na ekstrakciji nektara, a koji potom informacije o izvoru hrane dele sa ostalim pčelama. Za razliku od njih, cilj nezaposlenih tragaoca (unemployed foragers), koji mogu biti skauti-izviđači ili nadzornici-posmatrači (onlooker), sastoji se u konstantnoj potrazi za hranom ili u čekanju u košnici i deljenju

informacija koje dobijaju od zaposlenih tragaoca [48]. Informacije o izvoru hrane pčele tragaoci prenose drugim pčelama u košnici pomoću specifičnog plesa. Na početku, zadatak pčela izviđača je da nađu lokaciju gde se nalazi izvor hrane i informaciju o tome proslede zaposlenim pčelama i posmatračima, koji zatim vrše eksploataciju datog izvora. Kada se jedan izvor hrane u potpunosti iscrpi, zaposleni postaju izviđači i proces pronalaska novog izvora ponovo počinje. Inspirisano ovim procesom, razvijen je optimizacioni algoritam veštačke kolonije pčela (artificial bee colony, ABC), gde pozicija izvora hrane predstavlja jedno rešenje problema, a kvalitet hrane (bogatstvo nektarom) odgovara fitnes funkciji. Generalna struktura ABC algoritma je data u nastavku [49].

Algoritam 2.3: Generalna struktura ABC algoritma

```
1 begin
2     Faza inicijalizacije;
3     repeat
4         Faza zaposlenih pčela;
5         Faza pčela posmatrača;
6         Faza pčela izviđača;
7         Memorisati najbolje rešenje do tekućeg trenutka;
8     until dostignut maksimalni broj ciklusa ili maksimalno CPU vreme;
9 end
```

U fazi inicijalizacije, ABC algoritam prvo generiše uniformno raspodeljenu populaciju koju čini SN rešenja. Svako rešenje x_i $(i = 1, 2, \ldots, SN)$ je D-dimenzionalni vektor, gde je D broj promenljivih u datom optimizacionom problemu. Takođe, x_i predstavlja i-ti izvor hrane, koji se generiše na sledeći način [50]:

$$x_i^j = x_{min}^j + rand(0,1)(x_{max}^j - x_{min}^j), \forall j = 1, 2, \ldots, D \tag{2.13}$$

gde x_{max}^j i x_{min}^j predstavljaju granice rešenja x_i u dimenziji j.

U fazi zaposlenih pčela, pčela vrši modifikaciju tekućeg rešenja na osnovu svog individualnog iskustva i fitnes vrednost novog izvora hrane (rešenja). Ukoliko je fitnes vrednost (količina nektara) novog izvora hrane veća od starog izvora, pčela vrši ažuriranje svoje pozicije i odbacuje staru poziciju. Drugim rečima, primenjuje se mehanizam pohlepne selecije (greedy selection) kao operacija selekcije između starog i

novog, kandidatskog rešenja. Jednačina ažuriranja za j-tu dimenziju je:

$$v_{ij} = x_{ij} + \phi(x_{ij} - x_{kj}) \tag{2.14}$$

gde je: $\phi(x_{ij} - x_{kj})$ veličina koraka (step size); $k \in \{1, 2, \ldots, SN\}$ i $j \in \{1, 2, \ldots, D\}$ su dva slučajno izabrana indeksa, pri čemu i mora biti različito od k; ϕ_{ij} je slučajan broj iz intervala $[-1, 1]$.

Faza pčela posmatrača počinje kada zaposlene pčele podele informacije o fitnes vrednosti novih ažuriranih rešenja tj. izvora hrane i njihovim pozicijama sa pčelama posmatračima koje se nalaze u košnici. Pčele posmatrači imaju zadatak da analiziraju dobijene informacije i da izaberu rešenje u skladu sa verovatnoćom, koja je funkcija fitnes vrednosti rešenja. Postoji više načina za određivanje ove verovatnoće, a jedan od mogućih je prikazan sledećom formulom (fit_i je fitnes vrednost i-tog rešenja) [50]:

$$p_i = \frac{fit_i}{\sum_{i=1}^{SN} fit_i} \tag{2.15}$$

Pčela posmatrač vrši upoređivanje fitnes vrednosti odabranog rešenja i prethodnog rešenja koje ima u memoriji i na osnovu toga se radi eventualna modifikacija pozicije. Ako je fitnes vrednost novog rešenja veća od prethodnog, u memoriju se beleži i čuva nova pozicija, a stara se odbacuje. Ako se pozicija izvora hrane ne menja u toku nekoliko iteracija, pokreće se faza pčela izviđača jer se pretpostavlja da je izvor hrane iscrpljen i napušten. Kontrolni parametar koji određuje broj iteracija nakon kojeg dolazi do aktiviranja faze izviđača se naziva granica napuštanja ili granica (limit for abanndonment, limit). Tada ona zaposlena pčela koja je inicijalno našla napušteni izvor hrane postaje pčela izviđač, i tada ona zamenjuje napušteni izvor hrane sa novim izvorom x_i u skladu sa jednačinom (2.13), tj. napušteni izvor hrane se menja slučajno izabranim izvorom iz prostora pretraživanja [51]. Na osnovu iznetog zaključuje se da je faza eksploracije ABC algoritma određena fazom izviđača, a eksploatacija se obavlja u fazama zaposlenih pčela i posmatrača. Pseudokod ABC algoritma je dat u Algoritmu 2.4 [52].

ABC algoritam ima nekoliko kontrolnih parametara koje je potrebno fino podesiti da bi se postigle što bolje performanse, i to: broj izvora hrane, granica, ϕ_{ij} (težinski

Algoritam 2.4: Pseudokod ABC algoritma

```
1  begin
2      Inicijalizovati populaciju rešenja, x_i, i = 1, 2, ..., SN;
3      Evaluirati populaciju;
4      ciklus = 1;
5      repeat
6          Generisati nova rešenja v_i u fazi zaposlenih pčela koristeći jednačinu
             2.14 i evaluirati ih;
7          Primeniti proces pohlepne selekcije u fazi zaposlenih pčela;
8          Izračunati verovatnoće p_i za rešenja x_i po jednačini 2.15;
9          Generisati nova rešenja v_i u fazi pčela posmatrača, polazeći od x_i koja
             su izabrana u zavisnosti od p_i, i evaluirati ih;
10         Primeniti proces pohlepne selekcije u fazi pčela posmatrača;
11         U fazi pčela izviđača odrediti da li postoje napušteni izvori hrane, i
             ako postoje zameniti ih sa slučajno izabranim rešenjima x_i, na osnovu
             jednačine (2.13);
12         Memorisati najbolje rešenje do sada;
13         ciklus = ciklus + 1;
14     until ciklus = maksimalni broj ciklusa(MCN);
15 end
```

faktor za razliku između tekućeg izvora hrane i slučajnog generisanog izvora hrane), i verovatnoća p_i koja služi za selekciju u fazi pčela posmatrača. Na osnovu analize u [53] zaključuje se da veličina kolonije pčela treba da bude između 50 i 10 pčela. Takođe, vrednost parametra granica, koji određuje početak faze pčela izviđača i na taj način utiče na uspostavljanje balansa između eksploracije i eksploatacije, treba da bude jednaka vrednosti $\frac{veličina\ kolonije}{2} * D$. Kada je u pitanju parametar ϕ_{ij}, koji utiče na veličinu koraka, pa samim tim i na diverzitet procesa pretraživanja ABC algoritma, preporučuje se da on bude slučajna vrednost koja pripada uniformnoj raspodeli iz intervala $[-1, 1]$. Efekat kontrolnih parametra granica i veličina koraka (tj. faktor skaliranja) je proučavana i u [54]. Rezultati istraživanja su pokazali da je 1 najpogodnija vrednost za parametar veličina koraka, a 200 najbolja vrednost za parametar granica, i u slučaju unimodalnih i multimodalnih funkcija. Naravno, ove vrednosti su zavisne od dimenzije problema koji se rešava.

2.5 Algoritam svica

U svetu postoji nekoliko hiljada vrsta svitaca i većina od njih u procesu koji se naziva bioluminiscencija ima sposobnost da proizvodi kratku svetlost određene jačine i ritma, pri čemu karakteristike svetlosti zavise od vrste svica. Dve osnovne funkcije svetlosti koju emituju svici su da privuku potencijalnog partnera za parenje ili potencijalni plen. Takođe, svetlucanje može da bude i u cilju upozorenja i odbijanja drugih vrsta koje mogu da im naškode. Svetlost koju svici emituju i privlačenje između njih je inspirisalo Xin-Shi Yanga [55] da razvije algoritam za optimizaciju koji je nazvan algoritam svica (firefly algorithm, FA). U ovom algoritmu, svetlost koja se emituje je na pogodan način formulisana i povezana sa funkcijom cilja koju treba optimizovati. FA koristi tri idealizovana pravila:

- Svici su bespolni tako da će se oni međusobno privlačiti bez obzira na pol;
- Privlačnost je direktno proporcionalna intenzitetu svetlosti koja se smanjuje sa povećanjem rastojanja tako da u slučaju dva svica, svitac koji manje svetli će se kretati ka onom koji više svetli;
- Intenzitet svetlosti svica je određena karakteristikama funkcije cilja.

Osnovni koraci realizacije FA algoritma su dati pseudokodom u Algoritmu 2.5 [56].

U najjednostavnijem slučaju, jačina svetlosti I nekog svica na određenoj lokaciji x može biti definisana kao $I(x) \propto f(x)$. Međutim, privlačnost β je relativna, jer nju procenjuju ostali svici ili posmatrač. Dakle, privlačnost se menja sa udaljenošću r_{ij} između svica i i svica j. Poznato je da intenzitet svetlosti opada sa udaljenošću od izvora jer se deo svetlosti apsorbuje od strane medijuma za prenos, te prema tome, privlačnost se može posmatrati kao funkcija stepena apsorpcije. Ako usvojimo da intenzitet svetlosti opada sa kvadratom rastojanja od izvora, i ako koeficijent apsorpcije medijuma označimo sa γ, dobijamo formulu za izračunavanje intenziteta svetlosti [57]:

$$I = I_0 e^{-\gamma r} \tag{2.16}$$

Ako usvojimo da je privlačnost svica direktno proporcionalna intenzitetu svetlost

Algoritam 2.5: Algoritam svica

```
1  begin
2      Funkcija cilja f(x), x = (x_1, x_2, ..., x_d)^T;
3      Generisati inicijalnu populaciju svitaca, x_i, i = 1, 2, ..., n;
4      Intenzitet svetlosti I_i svica x_i je određena preko f(x_i);
5      Definisati koeficijent apsorpcije svetlosti, γ;
6      while t < MaksGenerisanja do
7          for i = 1 to n do
8              for j = 1 to n do
9                  if I_i < I_j then
10                     Pomeriti svica i ka svicu j
11                 end
12                 Varirati privlačnost sa rastojanjem pomoću exp[−γr];
13                 Evaluirati nova rešenja i ažurirati intenzitet svetlosti;
14             end
15         end
16         Rangirati svice i naći tekuće globalno najbolje rešenje (global best) g;
17     end
18     Post procesiranje i vizuelizacija;
19 end
```

koju vide susedni svici, onda se privlačnost svica može definisati sledećom jednačinom:

$$\beta = \beta_0 e^{-\gamma r^2} \tag{2.17}$$

gde je β_0 privlačnost u tački $r = 0$. U implementacji algoritma, funkcija privlačnosti može biti bilo koja monotono opadajuća funkcija, kao npr. sledeća generalizovana forma:

$$\beta(r) = \beta_0 e^{-\gamma r^m}, (m \geq 1) \tag{2.18}$$

Udaljenost između bilo koja dva svica i i j, na pozicijama x_i i x_j, respektivno, je data Dekartovim rastojanjem:

$$r_{ij} = \|x_i - x_j\| = \sqrt{\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2} \tag{2.19}$$

gde je $x_{i,k}$ k-ta komponenta prostorne koordinate x_i i-tog svica. U slučaju $2D$ sistema,

dobija se:

$$r_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{2.20}$$

Kretanje svica i koji je privučen od strane atraktivnijeg (svetlijeg) svica j je određeno sledećom jednačinom [58]:

$$x_i = x_i + \beta_0 e^{-\gamma r_{ij}^2}(x_j - x_i) + \alpha\epsilon_i \tag{2.21}$$

Drugi sabirak u jednačini (2.21) služi za određivanje uticaja privlačnosti, a treći sabirak je randomizacija, pri čemu je α parametar randomizacije, a ϵ_i je vektor slučajnih brojeva koji mogu da pripadaju Gausovoj ili uniformnoj raspodeli. U najjednostavnijoj formi, vektor slučajnih brojeva može da bude zamenjen izrazom $rand\,(-1/2)$, gde je *rand* generator slučajnih brojeva koji imaju uniformnu raspodelu iz intervala $[0, 1]$. Parametar α služi za kontrolu randomizacije, pa samim tim utiče i na diverzitet rešenja. On se može podesiti tako da se njegova vrednost menja tokom izvršavanja algoritma. Jedan od načina za definisanje ovog parametra je pomoću izraza:

$$\alpha = \alpha_0 \delta^t \tag{2.22}$$

gde je α_0 inicijalni skalirajući faktor randomizacije, a δ je faktor hlađenja. Obično se u implementaciji algoritma uzimaju vrednosti za δ od 0.95 do 0.97 [59]. Ako sa L obeležimo prosečan nivo (scale) problema od interesa, dobro je podesiti da α_0 bude $0.01L$. Parametar γ služi za definisanje varijacije privlačnosti, i njegova vrednost je veoma važna u određivanju brzine konvergencije FA algoritma. Pokazalo se da i vrednost ovog parametra treba podesiti u odnosu na L, i to kao $\gamma = 1/\sqrt{L}$. Obično je vrednost ovog parametra između 0.1 i 10. Takođe, uzima se da veličina populacije n bude od 15 do 100 [60].

Analizom jednačine (2.21) može se pokazati da se FA algoritam može svesti na neke druge optimizacione algoritme, tačnije da su algoritmi diferencijalne evolucije, ubrzani PSO algoritam, simulirano kaljenje i harmonijsko pretraživanje specijalni slučajevi FA algoritma. Na primer, ako je γ veoma veliko, onda drugi sabirak u jednačini (2.21) postaje mali, pa se algoritam svodi na algoritam simuliranog kaljenja. S druge strane, ako parametar γ ima malu vrednost (teži nuli) onda se dobija varijanta algoritma diferencijalne evolucije. Slično se može pokazati i za ostale navedene algoritme.

2.6 Kukavičje pretraživanje

Kukavice su ptice koje prilikom reprodukcije ispoljavaju parazitno ponašanje. Najveći broj vrsta to ispoljavaju putem polaganja jaja u gnezda drugih ptica. Pritom, one mogu da uklone jaja ptica domaćina kako bi povećali verovatnoću izleganja sopstvenih jaja ili kako bi hrana koja se nalazi u gnezdu bila na raspolaganju samo njenim mladuncima. Ako ptica domaćin otkrije kukavičja jaja, ona može da ih izbaci ili da napusti gnezdo [61]. Algoritam za optimizaciju koji se zasniva na ponašanju ptica kukavica naziva se kukavičke pretraživanje (cuckoo search, CS).

Da bi se algoritam uspešno implementirao uvode se tri jednostavna pravila [62]: 1) svaka kukavica polaže samo jedno jaje u datom trenutku u slučajno izabrano gnezdo; 2) najbolja gnezda sa najkvalitetnijim jajima (rešenjima) idu u sledeću generaciju; 3) broj gnezda na raspolaganju je fiksan, a verovatnoća da domaćin otkrije tuđe jaje je p_a, kada on ili napušta gnezdo i ide da pravi drugo, ili izbacuje uljeza. Pseudokod CS algoritma je prikazan u Algortimu 2.6 [63].

Algoritam 2.6: Kukavičje pretraživanje

```
1  begin
2      Funkcija cilja f(x), x = (x_1, x_2, ..., x_d)^T;
3      Generisati inicijalnu populaciju gnezda, x_i, i = 1, 2, ..., n;
4      while t < MaksGenerisanja ili uslov prestanka do
5          Na slučajan način, pomoću Levijevog leta izabrati kukavicu (npr. i);
6          Evaluirati njen kvalitet/fitnes F_i;
7          Na slučajan način izabrati gnezdo j (od mogućih n);
8          if (F_i > F_j) then
9              Zameni j sa novim rešenjem;
10         end
11         Napustiti deo (p_a) najgorih gnezda (i napraviti nova na novim
            lokacijama, pomoću Levijevog leta);
12         Zadržati najbolja rešenja (gnezda sa najboljim kvalitetom);
13         Rangirati rešenja i naći najbolje rešenje;
14     end
15     Post procesiranje i vizuelizacija;
16 end
```

Generisanje novog rešenja za kukavicu i se primenjuje pomoću Levijeg leta, u

skladu sa jednačinom (2.23):

$$x_i^{(t+1)} = x_i^{(t)} + \alpha \oplus Levy(\lambda) \tag{2.23}$$

gde je $\alpha > 0$ veličina koraka koja zavisi od veličine problema od interesa, a obično je jednaka 1. Jednačina (2.23) je stohastička jednačina za slučajni hod, koji uopšteno gledano predstavlja Markovljev lanac čiji je sledeći status/lokacija određena tekućom lokacijom (prvi sabirak) i tranzicionom verovatnoćom (drugi sabirak). Levijev let je način za realizaciju slučajnog hoda gde slučajna dužina koraka pripada Levy raspodeli: $Levy \sim u = t^{-\lambda}, (1 < \lambda \leq 3)$

2.7 Algoritam slepog miša

Mikro slepi miševi su uglavnom insektivori koji koriste eholokaciju da bi detektovali plen ili izbegli prepreku. Oni emituju veoma jake i kratke zvučne signale određene frekvencije i slušaju eho koji dolazi od objekata u okruženju. Zvučni signali obično traju nekoliko milisekundi, sa konstantnom frekvencijom u opsegu od 25kHz do 100kHz. Slepi miš emituje od 10 do 20 signala u sekundi, a kada juri plen emituje i do 200 signala u sekundi. Jačina emitovanog zvučnog signala ide i do 110dB, pri čemu je frekvencija veća ako juri plen, a manja kada se vraća iz lova [64]. Slepi miševi koriste vremensko kašnjenje između emitovanja signala i detekcije eha, vremensku razliku između dva uva, i varijacije u jačini eha kako bi konstruisali trodimenzionalnu sliku okruženja. Bat optimizacioni algoritam (bat algorithm, BA) je razvijen imitiranjem ponašanja slepih miševa, tačnije njihove sposobnosti eholokacije. U BA se koriste idealizovane karakteristike eholokacije, koje se mogu izraziti pomoću sledeća tri pravila [65]: 1) Svi slepi miševi koriste eholokaciju radi detekcije rastojanja, a takođe na neki „magičan način“ znaju i razliku između hrane/plena i prepreka u okruženju; 2) Slepi miševi, kada su u potrazi za hranom, lete brzinom v_i, na poziciji x_i, emitujući zvuk fiksne frekvencije f_{min}, promenljive talasne dužine λ i jačine A^0. Oni mogu automatski da podese tj. ažuriraju talasnu dužinu (ili frekvenciju) emitovanih zvučnih signala tj. impulsa, kao i da podese brzinu emitovanja impulsa $r \in [0, 1]$, zavisno od blizine mete; 3) pretpostavlja se da jačina zvučnih impulsa može da varira između velike (pozitivne) vrednosti A^0 do minimalne vrednosti A_{min}. Pseudokod BA algoritma je prikazan u

Algoritmu 2.7 [66].

Algoritam 2.7: Algoritam slepog miša

```
1  begin
2      Funkcija cilja f(x), x = (x_1, x_2, ..., x_d)^T;
3      Generisati inicijalnu populaciju slepih miševa, x_i, i = 1, 2, ..., n, i brzine v_i;
4      Definisati frekvenciju zvučnih signala f_i na poziciji x_i;
5      Inicijalizovati brzinu emitovanja pulseva r_i, i jačinu A_i;
6      while t < Maksimalan broj iteracija do
7          Generisati nova rešenja podešavanjem frekvencije i ažuriranjem brzina
            i lokacija/rešenja (jednačine (2.24)-(2.26));
8          if rand < r_i then
9              Izabrati rešenje između svih najboljih rešenja;
10             Generisati lokalno rešenje u okolini izabranog najboljeg rešenja;
11         end
12         Generisati novo rešenje postupkom slučajnog leta;
13         if (rand < A_i & f(x_i) < f(x*)) then
14             Prihvatiti novo rešenje;
15             Uvećati r_i i smanjiti A_i;
16         end
17         Rangirati slepe miševe i naći trenutno najbolje rešenje x*;
18     end
19     Post procesiranje i vizuelizacija;
20 end
```

Pravila za ažuriranje frekvencije, brzine i lokacije/rešenja su data jednačinama (2.24), (2.25) i (2.26), respektivno.

$$f_i = f_{min} + (f_{max} - f_{min}) * \beta \tag{2.24}$$

$$v_i^t = v_i^{t-1} + (x_i^t - x^*) * f_i \tag{2.25}$$

$$x_i^t = x_i^{t-1} + v_i^t \tag{2.26}$$

3 BRAIN STORM OPTIMIZACIONI ALGORITAM

Brain storm optimizacioni algoritam (BSO) je algoritam inteligencije rojeva koji je predložio Yuhui Shi 2011. godine. To je nova metoda za optimizaciju koja se bazira na kolektivnom ponašanju ljudskih bića prilikom rešavanja problema, tj. na brainstorming procesu [67]. Brainstorming proces je interpretiran i napisan kao algoritam od nekoliko koraka. Ovi koraci pojednostavljuju ceo proces, ali sadrže dovoljno elemenata da bi se implementirao optimizacioni algoritam koji ima eksploraciju i eksploataciju. Svi algoritmi inteligencije rojeva imaju jednostavne agente koji u više generacija razmenom informacija poboljšavaju globalno najbolja rešenja. Eksploatacija se odnosi na lokalno pretraživanje u oblastima koje obećavaju, dok je eksploracija slučajna pretraga koja se koristi kako bi se izbeglo zaglavljivanje algoritma u lokalnom optimumu.

Ovaj algoritam ima dva glavna operatora: divergentni i konvergentni operator, a dovoljno dobro suboptimalno rešenje se dobija rekurzivnom primenom divergencije i konvergencije u prostoru pretraživanja. Divergencija se odnosi na eksploraciju novih prostora pretraživanja, dok konvergencija predstavlja eksploataciju postojećih regiona u kojima se mogu nalaziti dobra rešenja. S obzirom da brain storm algoritam pripada grupi razvojnih algoritama optimizacije rojeva, on ima dve glavne mogućnosti: učenje o sposobnositma i razvoj kapaciteta. Divergentni operator odgovara učenju o sposobnostima, dok se konvergentni operator odnosi na razvoj kapaciteta. Osobina BSO algoritama je klasterovanje rešenja, pri čemu se nova rešenja generišu mutacijama klastera ili postojećih rešenja. Kao što je rečeno, BSO algoritam se bazira na brainstorming procesu koji se odvija kada grupa ljudi zajednički rešava neki problem. Tipični koraci u ovom procesu kao i Ozborn-ova pravila za generisanje ideja su prikazani u Tabelama 3.1 i 3.2, respektivno [68].

Pravilo 2 kaže da bilo koja ideja koja nam padne na pamet u brainstorming procesu je interesantna i ne treba je odbacivati, već je treba podeliti sa ostalima, pri čemu je zabranjeno bilo kakvo osuđivanje i označavanje ideja kao besmislenih i bezvrednih (Pravilo 1). Potrebno je generisati što više ideja (Pravilo 4) polazeći od trenutnih ideja i njihovih međusobnih kombinacija (Pravilo 3), i ponavljati proceduru dok se ne dođe do zadovoljavajućeg rešenja. Polazeći od prethodno definisanog brainstorming procesa i pravila koja se trebaju poštovati, napravljen je BSO optimizacioni algoritam čiji je pseudokod prikazan u Algoritmu 3.1 [68].

Algoritam 3.1: Brain storm algoritam

```
1  Initialization;
2  Metodom slučajnog izbora generisati n potencijalnih rešenja (individua);
3  repeat
4      Podeliti n individua u m klastera;
5      Rangirati individue u svakom klasteru i označiti najbolje individue u klasterima
         kao centre klastera;
6      Metodom slučajnog izbora generisati vrednost r između 0 i 1;
7      if r < p_5a then
8          Metodom slučajnog izbora odrediti centar klastera;
9          Metodom slučajnog izbora generisati individuu koja će da zameni izabrani
             centar klastera;
10     end
11     repeat
12         Generisati nove indivudue;
13         Metodom slučajnog izbora generisati vrednost r između 0 i 1;
14         if r < p_6b then
15             Metodom slučajnog izbora izabrati klaster sa verovatnoćom p_6bi;
16             Metodom slučajnog izbora generisati vrednost r_1 između 0 i 1;
17             if r_1 < p_6bii then
18                 Izabrati centar klastera i dodati mu slučajnu vrednost da bi se
                     generisala nova individua;
19             else
20                 Metodom slučajnog izbora izabrati individuu iz izabranog klastera i
                     dodati joj slučajnu vrednost da bi se generisala nova individua;
21             end
22         else
23             Metodom slučajnog izbora izabrati dva klastera da bi se generisala nova
                 individua;
24             Generisati slučajnu vrednost r_2 između 0 i 1;
25             if r_2 < p_6c then
26                 Dva centra klastera se kombinuju, pa im se dodaje slučajna vrednost
                     da bi se generisala nova individua;
27             else
28                 Dve individue iz svakog od izabranih klastera se biraju metodom
                     slučajnog izbora i zatim se kombinuju, pa im se dodaje slučajna
                     vrednost da bi se generisala nova individua;
29             end
30         end
31         Nove generisane individue se porede sa postojećim individuama sa istim
             individualnim indeksom, bolja se zadržava i beleži kao nova individua;
32     until n novih individua je generisano;
33 until dostignut maksimalni broj iteracija;
```

Tabela 3.1 Koraci u brainstorming procesu

Korak 1. Napraviti grupu od ljudi sa što većim razlikama u znanju i iskustvu;
Korak 2. Generisati što više ideja u skladu sa pravilima iz Tabele 3.2;
Korak 3. Izabrati nekoliko, 3 do 5 klijenata koji će biti vlasnici problema i uzeti nekoliko, npr. od svakog vlasnika po jednu, kao bolje ideje za rešenje problema;
Korak 4. Ideje generisane u koraku 3 uzeti sa većom verovatnoćom od ostalih ideja kao polazišta, i generisati nove ideje u skladu sa pravilima iz Tabele 3.2;
Korak 5. Neka vlasnici izaberu nekoliko boljih ideja kao što su učinili u koraku 3;
Korak 6. Slučajno izabrati objekat i iskoristiti njegove funkcije i izgled kao polazište, a zatim generisati još ideja u skladu sa pravilima iz Tabele 3.2;
Korak 7. Neka vlasnici izaberu nekoliko boljih ideja;
Korak 8. Dovoljno dobro rešenje će se dobiti na osnovu generisanih ideja i njihovih kombinacija.

Tabela 3.2 Osborn-ova izvorna pravila za generisanje ideja u brainstorming procesu

Pravilo 1. Zaustavi osuđivanje i kriticizam;
Pravilo 2. Bilo šta je u redu;
Pravilo 3. Kombinuj i gradi novo na osnovu postojećeg (Cross fertilize and Piggybacking);
Pravilo 4. Ići na kvantitet.

U brain storm optimizacionom algoritmu ideje su d-dimenzionalni vektori i predstavljaju jednostavne agente. Brainstorming proces počinje tako što se generiše n inicijalnih slučajnih rešenja, a nakon toga se ideje dele u m klastera. Za klasterovanje se obično koristi algoritam klasterovanja pomoću k-sredina, ali mogu se koristiti i ostale metode. U svakom klasteru se za centar klastera bira najbolja ideja, tj. ideja koja ima najbolju vrednost fitnes funkcije. U iterativnom procesu se kombinovanjem postojećih ideja (rešenja) kreiraju nova rešenja. Sa verovatnoćom p_{6b} se bira jedno rešenje $X^d_{selected}$ koje će biti izmenjeno da bi se generisalo novo rešenje. Nakon upoređivanja starog i novog rešenja, čuva se ono rešenje koje je bolje, tj. koje ima bolju vrednost fitnes funkcije.

Generisanje novih ideja (individua) iz koraka 6 se obavlja prema sledećoj formuli:

$$X^d_{new} = X^d_{selected} + \xi * n(\mu, \sigma) \tag{3.1}$$

gde je: X^d_{new} – d-ta dimenzija novokreirane individue; $X^d_{selected}$ – d-ta dimenzija individue koja je izabrana radi generisanja nove individue; $n(\mu, \sigma)$– funkcija Gauss-ove raspodele; a ξ – težinski koeficijent kojim se određuje doprinos Gauss-ove slučajne vrednosti. Ovaj koeficijent se naziva još i veličina koraka (step-size) i njegova vrednost se može izračunati na osnovu izraza:

$$\xi = logsig(\frac{0.5 * maxIteration - currentIteration}{k}) * rand() \tag{3.2}$$

gde je: $logsig()$ – logaritamska sigmoidna transfer funkcija; $maxIteration$ – maksimalni broj iteracija; $currentIteration$ – broj tekuće iteracije; k – vrednost koja menja nagib $logsig()$ funkcije; $rand()$ – slučajna vrednost iz intervala (0,1). Dodatno, sa verovatnoćom $1-p_{6b}$ će biti odabrana dva rešenja (individue) čijim će kombinovanjem biti generisano novo rešenje kao srednja vrednost odabranih rešenja. Proces eksploracije se kontroliše pomoću drugog algoritamskog parametra, p_{5a}, i implementira se kao zamena centra klastera novim slučajnim rešenjem. Osim pomenutih parametara, BSO algoritam ima još nekoliko parametara koji moraju biti određeni. Parametar p_{6bi} se odnosi na verovatnoću da će biti izabran klaster koji služi za selektovanje rešenja (individue) koja će biti korišćena za generisanje novog rešenja. Verovantoća p_{6bi} je proporcionalna broju ideja u svakom klasteru. Parametri p_{6bii} i p_{6c} su verovatnoće korišćenja centra klastera ili slučajnih rešenja iz izabranih klastera. Parametar p_{6bii} je verovatnoća ažuriranja centara klastera ili slučajnog rešenja iz klastera. Konačno, verovatnoća p_{6c} se koristi za određivanje da li će se kombinovati dva centra klastera ili dve slučajne ideje iz dva klastera.

3.1 Modifikacije BSO algoritma

Originalni BSO algoritam, opisan u prethodnoj sekciji, ima dve glavne komponente: konvergentni i divergentni operator. Konvergentni operator služi da emulira operaciju biranja boljih ideja od strane vlasnika problema tako da se na osnovu njih u narednoj rundi procesa generisanja ideja mogu kreirati nove ideje. Dakle, ovaj ope-

rator preslikava populaciju u jedan manji skup individua. Divergentni operator je zadužen za generisanje novih ideja na osnovu boljih ideja koje su odabrane u prethodnoj iteraciji. Sa aspekta rada algoritma, moguće je razlikovati tri različita operatora: operator grupisanja, operator zamene, i operator kreiranja. Operator grupisanja sve ideje generisane u toku jedne generacije grupiše u različite grupe (klastere), a operator kreiranja je zadužen da generiše nove ideje tako što dodaje slučajnu vrednost ideji iz jedne grupe ili kombinaciji ideja iz više grupa. Implementacija ovog algoritma kao operator grupisanja koristi metod klasterovanja pomoću k-sredina, a Gauss-ov slučajni šum se koristi kod operatora kreiranja kao slučajna vrednost koja se dodaje prilikom generisanja novih ideja. Ubrzo nakon predstavljanja originalnog BSO algoritma, javili su se predlozi i varijacije koje imaju za cilj poboljšanje njegovih performansi. Predložene su različite strategije i načini modifikacije operatora, odnosno parametara originalnog BSO, koji su rezultirali većom brzinom konvergencije i manjom vremenskom kompleksnošću predloženih algoritama. U nastavku je dat pregled značajnijih varijacija i modifikacija originalnog BSO algoritma.

Autori u [69] uvode dve nove modifikacije u originalni BSO algoritam kako bi poboljšali njegove performanse. Operacija kreiranja se modifikuje tako što se implementira dinamička strategija kontrole parametra veličina koraka. Operacija grupisanja se menja tako što se predlaže strategija slučajnog grupisanja umesto metoda k-sredina. S

Algoritam 3.2: Strategija slučajnog grupisanja

begin
- Slučajno podeliti N ideja u m grupa na osnovu veličine grupe $(N = s * m)$, $G = \{G_1, G_2, \ldots, G_m\}$;
- Za svaku grupu G_i, uporediti fitnes vrednosti svih individua u svakoj grupi;
- Izabrati centralnu ideju svake grupe kao onu koja ima minimalnu fitnes vrednost;
- Jednačina za podešavanje veličine koraka je:

$$\xi = rand() * \exp(1 - \frac{maxIteration}{maxIteration - currentIteration + 1}) \qquad (3.3)$$

  gde je $rand()$ slučajna vrednost između 0 i 1;

end

obzirom da je BSO stohastički optimizacioni algoritam, ova strategija povećava šanse za nalaženje dovoljno dobrih rešenja u heurističkom modu pretraživanja. Strategija slučajnog grupisanja se može predstaviti procedurom datom u Algoritmu 3.2.

Pseudokod RGBSO algoritma je dat u Algoritmu 3.3.

Algoritam 3.3: BSO algoritam sa slučajnim grupisanjem - RGBSO

```
1  begin
2      Slučajnim postupkom inicijalizovati N ideja i odrediti njihove fitnes
         vrednosti;
3      Inicijalizovati centre klastera (m < N);
4      while uslovi zaustavljanja nisu ispunjeni do
5          Izvršiti proceduru slučajnog grupisanja;
6          for i = 1 to N do
7              if rand() < p_one then
8                  if rand() < p_one_center then
9                      Izabrati centar grupe kao X_selected
10                 else
11                     Slučajno izabrati ideju iz grupe kao X_selected
12                 end
13             else
14                 if rand() < p_two_center then
15                     Kombinovati centre iz dve grupe da bi se formirao X_selected
16                 else
17                     Kombinovati dve slučajne ideje iz dve grupe da bi se
                         formirao X_selected
18                 end
19             end
20             Kreirati X_new pomoću X_selected koristeći formule (3.1) i (3.3);
21             Prihvatiti X_new ako je f(X_new) bolje od f(X_i)
22         end
23     end
24 end
```

Zhan i dr. [70] predlažu nove metode i strategije za operatore grupisanja i kreiranja. Predloženi modifikovani brain storm algoritam koristi jednostavni metod grupisanja (simple grouping method, SGM) za operator grupisanja, a strategiju razlikovanja ideja (idea difference strategy, IDS) za operaciju kreiranja. SGM metod se implementira procedurom prikazanom u Algoritmu 3.4. Inspiracija za IDS pristup dolazi od činjenice da u početku brainstorming procesa ljudi generišu različite ideje, dok ta različitost

Algoritam 3.4: SGM metod

begin

Metodom slučajnog izbora odabrati M različitih ideja iz tekuće generacije, kao semena M grupa. Ova semena se označavaju sa S_j,$(1 \leq j \leq M)$;

Za svaku ideju X_i,$(1 \leq i \leq M)$ iz tekuće generacije, izračunati njenu udaljenost do svake grupe po formuli:

$$d_g = \|X_i, S_j\| = \sqrt{\sum_{d=1}^{D} \frac{(x_i - s_j)^2}{D}}$$

;

Uporedi sve udaljenosti i dodeliti X_i najbližoj grupi;

Idi na korak 2. Ponavljati proceduru dok sve ideje ne budu dodeljene nekoj grupi;

end

opada kako se proces približava kraju. Dakle, prilikom kreiranja novih ideja na osnovu postojećih, mora se uzeti u obzir različitost trenutnih ideja. Ako sa Y_i obeležimo novu ideju, a sa X_a i X_b dve slučajno izabrane ideje koje treba da predstavljaju razliku ideja, onda se postupak kreiranja ideja po IDS metodi može predstaviti na sledeći način:

$$y_{id} = \begin{cases} random(L_d, H_d), & ako\ je \quad random(0,1) < p_r \\ x_{id} + random(0,1)_d * (x_{ad} - x_{bd}), & u\ suprotnom \end{cases} \tag{3.4}$$

Poboljšanje performansi BSO algoritma primenom mehanizma reinicijalizacije, kao i modifikacijom jednačine koraka tj. jednačine kojom se definiše težinski koeficijent za određivanje doprinosa slučajne vrednosti, se predlaže u [71]. Svakoj ideji se dodeljuje poseban brojač koji se inkrementira svaki put kada ideja nije poboljšana. Ako brojač pređe određenu vrednost ideja se reinicijalizuje u prostoru pretraživanja. Proces reinicijalizacije (Algoritam 3.5) se može obaviti na dva različita načina: 1) slučajna reinicijalizacija ideje u prostoru pretraživanja; 2) kombinacija tri slučajno izabrane ideje kako bi se generisala nova korišćenjem jednačine ažuiranja diferencijalne evolucije. Druga modifikacija se sastoji u promeni jednačine kojom se izračunava parametar ξ. Predlaže se uvođenje multiplikatora α, koji je proporcionalan veličini prostora pre-

Algoritam 3.5: Procedura reinicijalizacije

```
1  begin
2      for i = 1 to NumberofIdeas do
3          if counters(i) ≥ threshold then
4              if rand < 0.5 then
5                  Slučajno izabrati tri različite ideje j, k, l
                     idea(i) = idea(j) + F * (idea(k) − idea(l))
6              else
7                  idea(i) = LB + rand(1, D) * (UB − LB)
8              end
9          end
10     end
11     return nove ideje;
12 end
```

traživanja, tako da nova jednačina ima sledeći oblik:

$$\xi = rand * e^{\frac{1-MaxIteration}{MaxIteration-CurrentIteration+1}} * \alpha \tag{3.5}$$

Primena globalno-najboljeg koncepta u kombinaciji sa ažuriranjem na nivou promenljivih i grupisanjem na osnovu pogodnosti (fitness based grouping) radi unapređenja BSO algoritma je data u [72]. Predloženi algoritam (globalno-najbolji brain storm optimizacioni algoritam, GBSO) sadrži i reinicijalizacioni mehanizam koji je aktiviran trenutnim stanjem populacije. Koncept globalno-najboljih informacija je pozajmljen iz PSO, a ovde se u jednačini ažuriranja primenjuje na sledeći način:

$$nidea(i) = nidea(i) + rand(1, DimSize) * C * (GlobalBest - nidea(i)) \tag{3.6}$$

pri čemu je C dato sa:

$$C = C_{min} + \frac{CurrentIteration}{MaxIteration} * (C_{max} - C_{min}) \tag{3.7}$$

Za razliku od originalnog BSO algoritma, gde se nove ideje generišu tako što se istovremeno ažuriraju sve promenljive u jednom koraku, ovde se promenljive ažuriraju jedna po jedna. To znači da se prva promenljiva problema može ažurirati koristeći centar jednog slučajno izabranog klastera, sledeća promenljiva se ažurira na osnovu

kombinacije ideja iz dva slučajno izabrana klastera itd. Takođe, dok se u originalnom BSO za generisanje nove ideje koriste jedna ili najviše dve postojeće ideje, u ovom algoritmu se dozvoljava kombinacija više ideja, što omogućava jaču kooperaciju između individua.

Promene u ažuriranju veličine koraka i načina generisanja novih individua radi poboljšanja performansi originalnog BSO algoritma su date u [73]. Predloženi algoritam predviđa da veličina koraka bude adaptivna i da se ažurira u skladu sa dinamičkim opsegom individua u svakoj pojedinačnoj dimenziji. Veličina koraka, koja definiše uticaj Gauss-ovog šuma se može u modifikovanom algoritmu predstaviti sledećim jednačinama:

$$\xi_i^{center} = k_l * (x_{nmax,i} - x_{nmin,i}) \tag{3.8}$$

$$\xi_i^{individual} = k_2 * (x_{nmax,i} - x_{nmin,i}) \tag{3.9}$$

gde su ξ_i^{center} i $\xi_i^{individual}$ vrednosti veličine koraka u i-toj dimenziji i koriste se u različitim koracima algoritma, k_1 i k_2 su koeficijenti, a $x_{nmax,i}$ i $x_{nmin,i}$ su maksimalna i minimalna vrednost u celoj populaciji u i-toj dimenziji, respektivno. Na osnovu jednačina (3.8) i (3.9) može se uočiti da je veličina koraka adaptivna i da se prilagođava dinamičkom opsegu populacije individua. Takođe, predlaže se da se individue generišu u batch modu, a onda da se vrši selekcija za sledeću generaciju. Na ovaj način, algoritam kreira više individua kako bi se u potpunosti iskoristila svaka referentna tačka umesto da se stalno kreira isti broj individua koji odgovara veličini populacije. Delovi algoritma koji su izmenjeni u odnosu na originalni BSO, a koji se odnose na generisanje individua i njihovu selekciju dati su u Algoritmu 3.6.

Xue i dr. [74] predlažu algoritam za višeciljnu optimizaciju baziran na brainstorming procesu (MOBSO), koji se sastoji iz šest delova, pri čemu su tri dela specifična za BSO algoritme: strategija klasterovanja, proces generisanja, i ažuriranje globalne arhive. Uz standardne operacije koje se primenjuju u tradicionalnim višeciljnim optimizacionim algoritmima, u predloženom algoritmu se strategija klasterovanja primenjuje u prostoru ciljeva (objective space), pri čemu se primenjuje algoritam klasterovanja pomoću k-sredina da bi se izvršilo klasterovanje populacije u k klastera na osnovu svakog cilja. Pseudokod strategije klasterovanja je dat u Algoritmu 3.7. Nakon klasterovanja, generišu se nove ideje u skladu sa procesom odabiranja, korakom disperzije, kao i operatorima selekcije i mutacije. Operator mutacije generiše nova

Algoritam 3.6: Modifikacije BSO algoritma

begin
  Generisanje individua
  Slučajno izabrati centar klastera. Verovatnoća odabiranja bilo kojeg centra klastera je u skladu sa veličinom klastera. Na osnovu izabranog centra, generisati novu individuu korišćenjem jednačine (3.9). Ponavljati ovaj korak n puta (gde je n veličina populacije) i tako generisati n individua;
  Slučajno izabrati dva centra klastera, i spojiti ih kako bi se napravio novi centar. Na osnovu novog centra generisati novu individuu korišćenjem jednačine (3.8). Ponavljati ovaj korak n puta i tako generisati n individua;
  Slučajno izabrati individuu kao referentnu tačku. Dodati joj Gaussov šum primenom jednačine (3.9) kako bi se kreirala nova individua. Ponavljati ovaj korak n puta i tako generisati n individua;
  Selekcija
  Evaluirati nove generisane individue, kojih ima ukupno $3n$;
  Na slučajan način sortirati ukupno $4n$ individua u n grupa sa jednakim brojem individua (po 4). U svakoj grupi izabrati individuu koja ima najbolju fitnes vrednost i kopirati je u sledeću generaciju.
end

Algoritam 3.7: Strategija klasterovanja

begin
  Inicijalizuj $Elite\_set = \emptyset$, $Normal\_set = \emptyset$;
  Evaluiraj populaciju i ažuriraj $Archive\_set$ u skladu sa Pareto dominacijom;
  Inicijalizuj centre klastera na osnovu fitnes vrednosti cilja M;
  Za svaki cilj f_m: klasterovati populaciju u k klastera, izabrati klaster koji ima najbolju fitnes vrednost i označiti ga kao $Elite\_cluster_m$;
  Za svaku individuu: ako je individua bilo koja iz $Elite\_cluster_m$, onda je dodaj u $Elite\_set$; u suprotnom, dodati individuu u $Normal\_set$.
end

rešenja na osnovu postojećih, dok operator selekcije služi za odlučivanje da li će nova generisana ići u sledeću generaciju. Proces generisanja nove individue je prikazan u Algoritmu 3.8. Operacija mutacije se može realizovati kao Gauss-ova ili Cauchy-jeva mutacija. U klasičnim evolutivnim algoritmima se obično primenjuje ova prethodna, i data je jednačinama (3.1) i (3.2), dok se pokazalo da je Cauchy-jeva mutacija efikasan operator pretraživanja za veliki broj multimodalnih optimizacionih problema. Ova

Algoritam 3.8: Proces generisanja nove individue

```
1  begin
2      if rand() < P1 then
3          if rand() < P2 then
4              if rand() < P3 then
5                  slučajno izabrati individuu iz skupa Elite_set i označiti je kao
                     X_selected
6              else
7                  slučajno izabrati X_selected iz skupa Normal_set
8              end
9          else
10             slučajno izabrati X_selected iz skupa Archive_set
11         end
12     else
13         korak disperzije: slučajno generisati individuu X_selected
14     end
15 end
```

mutacija se može predstaviti sledećom jednačinom:

$$X_{new}^{d} = X_{selected}^{d} + \xi * C(\mu, \sigma) \tag{3.10}$$

gde je $C(\mu, \sigma)$ Cauchy-jeva funkcija sa srednjom vrednošću μ i varijansom σ. Operator selekcije se bazira na Pareto dominaciji i može se opisati na sledeći način: ako X_{new} dominira nad $X_{selected}$, onda X_{new} ide dalje; ako $X_{selected}$ dominira nad X_{new} ,onda $X_{selected}$ ide dalje; ako ne dominiraju jedan nad drugim, onda slučajnim izborom između njih odabrati novu individuu. Rezultati su pokazali da obe verzije algoritma (Gauss-ova i Cauchy-jeva) pokazuju odlične performanse.

U originalnom BSO algoritmu i mnogim njegovim modifikacijama se koristi metod klasterovanja pomoću k-sredina kao operator grupisanja. Mane ovog pristupa su to što on zahteva da broj klastera bude specificiran pre pokretanja algoritma iako tačan broj klastera ne može biti unapred poznat. Takođe, iako se broj klastera menja u toku rada BSO algoritma, metod k-sredina koristi fiksan broj klastera u toku celog iterativnog procesa. Autori u [75] predlažu da se umesto metode klasterovanja pomoću k-sredina koristi metod klasterovanja baziran na propagaciji afiniteta koji ne zahteva da broj klastera bude unapred određen ili procenjen pre pokretanja algoritma. Algori-

tam propagacije afiniteta je novi metod za klasterovanje zasnovan na tehnici razmene poruka. Ukratko, ako je dat skup tačaka $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ i funkcija s koja izračunava sličnost između dve tačke, onda važi da je $s(i,j) > s(i,k)$ akko α_j sličnija α_i od α_k. Algoritam se izvršava tako što se obavljaju dva koraka razmene poruke da bi se ažurirale dve matrice, matrica odgovornosti i matrica dostupnosti i na kraju pronašli egzemplari, tj. članovi ulaznog skupa koji su reprezenti klastera. Vrednost $r(i,k)$ iz matrice odgovornosti određuje koliko je α_k pogodan da služi kao egzemplar za α_i. Vrednost $a(i,k)$ iz matrice dostupnosti kvantifikuje koliko je pogodno da α_i izabere α_k kao svoj egzemplar. Takođe se uvodi i operator kreiranja bez argumenta koji se zasniva na neodređenim, distribuiranim informacijama od više klastera primenjujući ideje iz cloud drops algoritma. Preciznije, bira se više klastera koji se zatim kvantifikuju kao tri numeričke karakteristike: očekivana vrednost (Ex), entropija (En) i hiperentropija (He). Nova rešenja se generišu na osnovu ovih numeričkih karakteristika.

Autori u [76] predlažu poboljšani BSO algoritam koji koristi strategiju klasterovanja na bazi propagacije afiniteta i poboljšani operator kreiranja koji koristi strukturne informacije dobijene od jednog ili više klastera. Radi lakšeg izvlačenja informacija, fitnes vrednosti kandidatskih rešenja se mapiraju u uniformni interval poverenja. Zatim se izvlače strukturne informacije za svaki klaster C, i označavaju kao vektor $C\{u, v, w\}$, gde je u kernel klastera, a v i w su pokrivenost i disperzija kandidatskih rešenja, respektivno. Algoritam 3.9 sadrži pseudokod procedure za izvlačenje strukturnih informacija za svaki klaster. U slučaju više klastera, strukturne informacije se mogu dobiti primenom odgovarajućih formula. Kreiranje novih individua se na kraju zasniva na korišćenju različitih strukturnih informacija. Pseudokod za kreiranje novih individua je dat u Algoritmu 3.10.

Zhu i Shi [77] su uočili da je algoritam za klasterovanje k-sredinama, koji se koristi u originalnom BSO algoritmu, pogodan za programiranje i izračunavanja. Međutim, on ima i određene mane koje mogu da utiču na efikasnost algoritma. Naime, prilikom ažuiriranja centra klastera, na taj proces mogu negativno da utiču određeni autlajeri, zato što tada svaka individua izračunava srednju vrednost. Takođe, kada je skup podatak veliki, algoritam k-sredina može da zahteva veoma veliko vreme izračunavanja. Da bi se rešili ovi problemi, autori predlažu novi algoritam za klasterovanje koji se zasniva na korišćenju medijane umesto srednje vrednosti. Osnovna razlika između

Algoritam 3.9: Algoritam za izvlačenje strukturnih informacija

begin
  Izabrati klaster sa K individua u M dimenzija;
  Izračunati fitnes vrednosti svake individue: $f(x), i = 1, 2, \ldots, K$;
  Mapirati fitnes vrednosti u interval poverenja $[0, 1]$;
  Odrediti najbolju individuu koja predstavlja kernel u izabranom klasteru $u = [u_1, u_2, \ldots, u_M]$;
  Izračunati pokrivenost kao: $v = [v_1, v_2, \ldots, v_M] = \frac{1}{L-1} \sum_{i=1}^{L} (x_i - u)^2$;
  Za svaki par koji čine individua i njena fitnes vrednost, $(x_i, f(x_i))$ izračunati $o_i = \sqrt{\frac{-(x_i-u)^2}{2 \ln f(x_i)}}$;
  Izračunati srednju vrednost o_i kao o_{sr}, $o_{sr} = \frac{1}{L} \sum_{i=1}^{L} o_i$;
  Izračunati disperziju izabranog klastera na sledeći način:
  $w = [w_1, w_2, \ldots, w_M] = \frac{1}{L-1} \sum_{i=1}^{L} (o_i - o_{sr})^2$;
end

Algoritam 3.10: Algoritam za kreiranje novih individua

begin
  Izabrati klaster sa K individua u M dimenzija;
  Kreirati novih G individua na osnovu strukturnih informacija $C(u, v, w)$;
  for $j = 1$ **to** G **do**
    Generisati vektor p normalne raspodele pomoću kernela v i pokrivenosti w, kao $p = NormRand(v, w)$;
    Generisati nove indivudue x_i sa normalnom raspodelom, pomoću kernela u i pokrivenosti p, kao $x_i = NormRand(u, p)$;
  end
end

ova dva algoritma je u procesu ažuriranja centra klastera; kod novog algorima centar klastera postaje medijana, a ne srednja vrednost individua. Koordinata medijane u višedimenzionalnom prostoru je medijana u svakoj pojedinačnoj dimenziji.

Shi [78] predlaže modifikaciju konvergentnog operatora tako da se on implementira u 1-dimenzionalnom prostoru ciljeva umesto u prostoru rešenja tako da će vreme izračunavanja zavisiti od veličine populacije, ali ne i od dimenzije problema. Autor

polazi od stanovišta da iako se metod klasterovanja pokazao dobrim za realizaciju konvergentnog operatora, on nije neophodan za njegovu realizaciju, već to može da bude bilo koji metod koji omogućuje odabiranje boljih ideja iz populacije. On predlaže konvergentni operator u jednodimenzionalnom prostoru ciljeva (objective space) za primenu kod BSO koji rešava jednociljne optimizacione probleme, na isti način kao što je to primenjivano kod rešavanja višeciljnih optimizacionih problema u [74]. Procedura BSO u prostoru ciljeva je prikazana u Algoritmu 3.11.

Algoritam 3.11: Procedura BSO u prostoru ciljeva

```
1 Inicijalizacija populacije;
2 while nije kraj do
3     Evaluiranje individua;
4     Uzeti prvih perc_e procenata individua kao elitne, a ostale kao normalne;
5     Narušiti (disrupt) slučajno izabranu individuu;
6     Ažuriranje individua;
7 end
8 Izlaz individua;
```

Koraci 3 i 6 su isti kao u originalnom BSO algoritmu. Korak 4 predloženog algoritma zamenjuje operaciju klasterovanja, a korak 5 menja operaciju narušavanja (zamene) klastera u originalnom BSO. Korak 4 radi rangiranje individua u skladu sa njihovom fitnes vrednošću. Za razliku od originalnog BSO gde se individue grupišu u m klastera, ovde se najboljih $perc_e$ % individua smešta u kategoriju elitnih, dok se ostatak smešta u normalne. Kada je reč o ažuriranju (generisanju) novih individua, primenjuju se iste formule kao kod originalnog BSO, ali se prvo odlučuje da li će se nove individue generisati na osnovu elitnih ili normalnih individua, a zatim da li će se koristiti jedna ili dve odabrane individue (iako se, uopšteno gledano, svaka nova individua može kreirati i na osnovu više od dve odabrane individue). Pseudokod za ovu operaciju je dat u Algoritmu 3.12 (p_e - verovatnoća da će se prilikom generisanja novih individua koristiti elitne, a ne normalne individue; p_{one} - verovatnoća da će prilikom generisanja nove individue koristiti jedna, a ne dve individue).

Korak 5, koji se odnosi na operaciju narušavanja slučajno odabrane promenljive se obavlja tako da se narušava vrednost samo jedne, slučajno odabrane dimenzije date odabrane individue, gde se data vrednost menja slučajno generisanom vrednošću. Ovo se radi da bi se smanjila slučajnost koju unosi operacija narušavanja, koja se kod

Algoritam 3.12: Pseudokod za generisanje novih individua

```
1  if rand < p_e then
2      if rand < p_one then
3          generisati novu individuu na osnovu slučajno odabrane elitne individue
4      else
5          generisati novu individuu na osnovu dve slučajno odabrane elitne
             individue
6      end
7  else
8      if rand < p_one then
9          generisati novu individuu na osnovu slučajno odabrane normalne
             individue
10     else
11         generisati novu individuu na osnovu dve slučajno odabrane normalne
             individue
12     end
13 end
```

originalnog BSO obavlja tako da se odabrana individua menja slučajno generisanom individuom. Radi kompenzacije, korak 5 se izvršava u svakoj iteraciji, umesto u svakoj iteraciji sa određenom verovatnoćom, kao kod originalnog BSO.

Većina algoritama inteligencije rojeva ima manu preuranjene konvergencije. To se dešava između ostalog i zbog toga što se rešenja relativno brzo grupišu u male regione, što znači da se diverzitet populacije brzo smanjuje u toku pretraživanja tako da je otežana dalja divergencija. Ovo može da rezultuje u narušavanju balansa između eksploracije i eksploatacije, i da se algoritam zaglavi u lokalnom optimumu. Predlog da se mana prevaziđe je primena dve strategije reinicijalizacije rešenja koje treba da povećaju diverzitet populacije [79]. Generalna ideja je da se nakon nekoliko iteracija primeni parcijalna reinicijalizacija rešenja tj. da deo rešenja reinicijalizuje svoj položaj i brzinu u celom prostoru pretraživanja, što povećava verovatnoću da rešenja izađu iz lokalnog optimuma. Analiziraju se dve strategije: 1) da se pola rešenja reinicijalizuje nakon određenog broja iteracija; 2) da se broj reinicijalizovanih rešenja smanjuje tokom procesa pretraživanja tako da se više od pola rešenja reinicijalizuje na početku pretraživanja, a da se broj reinicijalizovanih rešenja linearno smanjuje prilikom svake reinicijalizacije. Algoritam 3.13 prikazuje pseudokod modifikovanog BSO algoritma koji primenjuje postupak reinicijalizacije.

Algoritam 3.13: Modifikovani BSO algoritam koji primenjuje postupak reinicijalizacije

```
1  begin
2      Inicijalizacija: Na slučajan način generisati n potencijalnih rešenja
         (individua) i evaluirati ih;
3      while nije nađeno dovoljno dobro rešenje ili nije dostignut unapred
         određeni maksimalni broj iteracija do
4          Klasterovanje: Klasterovati n individua u m klastera pomoću
             algoritma za klasterovanje;
5          Generisanje nove individue: slučajno odabrati jedan ili dva klastera
             radi generisanja nove individue;
6          Selekcija: Nova generisana individua se poredi sa postojećom koja ima
             isti indeks individue, bolja se zadržava i beleži kao nova individua;
7          Reinicijalizacija: uraditi parcijalnu reinicijalizaciju nekih rešenja posle
             određenog broja iteracija;
8          Evaluirati n individua;
9      end
10 end
```

Problem preuranjene konvergencije koji može da izazove zaglavljivanje rešenja u lokalnom optimumu kod originalnog BSO algoritma se može rešiti i primenom haotične operacije [80], zahvaljujući osobinama haosa kao što kao što su ergodicitet, sopstvena stohastička osobina i osetljivost na inicijalne uslove. Originalni BSO je potrebno izmeniti tako da, nakon ažuriranja individua, jedna slučajno odabrana dimenzija jednog slučajno odabranog klastera bude ažurirana u haos modu, pri čemu je opseg haotičnog kretanja ceo prostor pretraživanja. Specifični dinamički model jednostavne haotične mape je dat jednačinom [81]:

$$x_{+1} = rx(1 - x) \tag{3.11}$$

gde je x vrednost iz opsega $(0, 1)$, r parametar koji kontroliše ponašanje haotične mape. Ako je $r = 4$, x postaje u potpunosti haotično u intervalu $(0, 1)$. S obzirom da je inicijalna vrednost haotičnog procesa u opsegu $(0, 1)$, trenutna pozicija se transformiše u vrednost iz ovog opsega, što se dobija primenom jednačine (3.12):

$$fposi = \frac{(cposi - lbound)}{(ubound - lbound)} \tag{3.12}$$

gde je: $fposi$ - pozicija u frakcionalnoj formi, $cposi$ - trenutna pozicija u prostoru pretraživanja, $lbound$ i $ubound$ - donja i gornja granica u prostoru pretraživanja, respektivno. Nakon završetka haotične operacije, frakcionalna forma se transformiše u poziciju u prostoru pretraživanja po formuli:

$$newposi = fposi * (ubound - lbound) + lbound \tag{3.13}$$

Nova pozicija dobijena po formuli (3.13) se zatim evaluira, i ako je njena fitnes vrednost bolja od fitnes vrednosti prethodne pozicije, onda se vrši zamena.

3.2 Hibridni algoritmi

Hibridni algoritam za rešavanje kontinualnih optimizacionih problema, baziran na BSO i algoritmu simuliranog kaljenja je dat u [82]. Predloženi algoritam integriše proces simuliranog kaljenja u brain storm optimizacioni algoritam. Deo koji se integriše je zadužen za kreiranje novih individua u kasnijim fazama evolutivnog procesa, menjajući operator kreiranja BSO algoritma. Tačnije, novi algoritam integriše SA proces u BSO u serijskom hibridnom modu, što znači da se SA proces izvršava nakon svakog obnavljanja populacije kako bi se nova, ažurirana populacija učinila stabilnom. Na taj način se smanjuje nestabilnost koju unosi slučajnost operatora kreiranja BSO algoritma.

U [83] se dizajnira hibridni samoadaptivni algoritam za efikasno pretraživanje u multidimenzionalnom domenu, koji predstavlja kombinaciju principa učenja iz BSO algoritma i optimizacionog algoritma baziranog na učenju (teaching-learning based optimization algorithm, TLBO). TLBO se zasniva na interakcijama između učenika i učitelja u nekom razredu, kao i među samim učenicima. Učenik koji ima najveće znanje u datom trenutku se smatra kao da je učitelj. Ozbornovo pravilo 3 (Tabela 3.2), kao ključni mehanizam brainstorming procesa, je primenjeno u TLBO tako da se ideje generisane u fazama učenja i predavanja međusobno kombinuju gradeći nove ideje. Hibridni algoritam se sastoji iz četiri faze: inicijalizacija, faza predavanja, faza učenja, i brainstorming faza. Inicijalizacija se obavlja pomoću matrice dimenzija MxN, gde je M veličina populacije, a N dimenzija problema koji se rešava. U fazi predavanja, prosek razreda se nalazi tako što određuje srednja vrednost svake dimenzije, pa se one

kombinuju odgovarajućom formulom. Učitelj u tekućoj iteraciji je najbolje rešenje date populacije. Kod višeciljnih problema, najbolja individua se nalazi tako što se primenjuje pogodan nedominirajući metod sortiranja. Jednačina koja opisuje proces mutacije u ovoj fazi sadrži faktor učenja koji se može učiniti adaptivnim uvođenjem slučajne vrednosti u formulu za njegovo izračunavanje, što algoritam čini samoevoluirajućim. U fazi učenja učenici prolaze kroz proces poboljšavanja i unapređivanja pomoću diferencijalne mutacije, gde vremenski gradijent koji postoji između učenika biva korišćen za realizaciju procesa mutacije. U brainstorming fazi se dobijene populacije iz prethodnih faza kombinuju, a zatim se primenjuje operator mutacije realizovan preko nelinearnih funkcija, kao u originalnom BSO.

Hibridni algoritam koji integriše procedure mutacije i ukrštanja DE algoritma u interklasterne i intraklasterne operatore kreiranja BSO algoritma je dat u [84]. Razlog za primenu DE strategije je to što je ona uglavnom bazirana na informacijama o udaljenosti i smeru i prednost joj je što nema pristrasnost u određenom pravcu. Operacija mutacije hibridnog algoritma se realizuje u skladu sa strategijom mutacije klasičnog DE algoritma, tj. strategija diferencijalne evolucije se dodaje na normalne ideje kako bi se kreirale nove, različite ideje. DE operatori hibridnog algoritma su intraklasterni operator diferencijalne evolucije i interklasterni operator diferencijalne evolucije. Intraklasterni operator se može predstaviti jednačinom (3.14), kojom se pokazuje da se nova ideja generiše pomoću razlike dve slučajno izabrane ideje iz jednog klastera, i centra klastera.

$$X_{new} = X_{center} + F * (X_{r1} - X_{r2}) \tag{3.14}$$

gde je F skalirajući faktor mutacije koji utiče na diferencijalnu varijaciju između dve ideje, a indeksi $r1$ i $r2$ su međusobno isključivi celi brojevi slučajno izabrani u datom izabranom klasteru. Nakon ovoga se primenjuje operacija ukrštanja algoritma DE kako bi se generisala nova rešenja. Interklasterni operator se može predstaviti jednačinom (3.15), gde se vidi da se nova ideja generiše pomoću razlike dve slučajno izabrane ideje iz dva različita klastera i globalno najbolje ideje (za sve klastere).

$$X_{new} = GlobalIdea + F * (X_{r1} - X_{r2}) \tag{3.15}$$

gde je $GlobalIdea$ najbolja ideja u svim klasterima, a X_{r1} i X_{r2} su normalne ideje

izabrane iz dva različita klastera. Nakon ovoga se primenjuje operacija ukrštanja algoritma DE kako bi se generisala nova rešenja. Takođe, radi efikasnije kontrole brzine konvergencije, predloženi algoritam uvodi novi način za izračunavanje veličine koraka:

$$\xi = rand * e^{1-\frac{maxIteration}{maxIteration-currentIteration+1}} \tag{3.16}$$

3.3 Višeciljni i multimodalni BSO algoritmi

Originalni BSO algoritam je namenjen rešavanju jednociljnih optimizacionih problema. Međutim, razvijene su modifikacije i varijante koje se mogu koristiti za rešavanje višeciljnih i multimodalnih optimizacionih problema. Kao što je poznato, višeciljni optimizacioni problemi nemaju jedinstveno rešenje, već je u pitanju skup kandidatskih rešenja, tako da nijedno rešenje nije bolje od bilo kojeg drugog iz tog skupa, u odnosu na definisane ciljeve. Ovaj skup se naziva Pareto-optimalni skup, a pridruženi vektori ciljeva formiraju površ u prostoru ciljeva, koja se naziva Pareto-front. Jedan od glavnih izazova višeciljnih optimizacionih problema je to što je veoma teško dobiti dovoljan broj nedominiranih rešenja koja odgovaraju regionu kolena (knee region) Pareto fronta, koji predstavlja maksimalni kompromis između ciljeva. Višeciljni BSO algoritam zasnovan na estimaciji u regionu kolena i klasterovanju u prostoru ciljeva [85] se može primeniti da bi se dobila tačka kolena (knee point) Pareto-optimalnog fronta. Za razliku od originalnog BSO, ovde se primenjuje strategija klasterovanja direktno u prostoru ciljeva umesto u prostoru rešenja, što ubrzava proces pronalaska Pareto-dominantnih oblasti u sledećoj iteraciji. Klasterovanje se vrši koristeći metod k-sredina, pri čemu se prvo generiše inicijalna distribucija k klaster centara u prostoru ciljeva. Zatim se svaka individua pridruži klasteru čiji joj je centar najbliži, u skladu sa njenom fitnes vrednošću. Na kraju se i centri klastera ažuriraju tako da centar klastera bude centar mase svih čestica koje pripadaju tom klasteru. Klasteri bez nedominiranih rešenja se mapiraju u prostor promenljivih odlučivanja (decision variable space) i sačinjavaju novu populaciju $Q1$. Operacija mutacije se primenjuje ako postoje klasteri bez nedominiranih rešenja. Tada se mutacija koristi u novoj populaciji kako bi se generisala rešenja umesto individua čija pozicija odstupa od nedominiranih rešenja. Mutacija je implementirana kao operacija mutacije iz diferencijalne evolucije,

i može se predstaviti sledećom jednačinom:

$$Z_i = P_i + F * (P_m + P_n) \tag{3.17}$$

gde je: Z_i – vektor pokušaja (trial vector), P_i – ciljni vektor (target vector), F - parametar mutacije, P_m i P_n – vektori parametara slučajno izabrani među nedominiranim rešenjima koja su međusobno različita. Verovatnoća mutacije se menja u skladu sa brojem iteracija kako bi se poboljšala konvergencija algoritma. Operacija selekcije tj. odlučivanje da li će nova generisana rešenja ići u sledeću generacija se zasniva na Pareto dominaciji. S obzirom da se pozicija tačke kolena menja nakon više iteracija, potrebno je da se uradi estimacija oblasti kolena. Algoritam za estimaciju regiona kolena bira nekoliko najboljih rešenja iz μ matrice da bi se kreirala kandidatska rešenja. Ako je euklidsko rastojanje između dve potencijalne tačke kolena manje od ϵ, ova dva rešenja su locirana u jednoj tački kolena koja se procenjuje primenom metoda srednje vrednosti; u suprotnom, dve potencijalne tačke kolena su dve različite tačke kolena. Takođe, primenjuje se i lokalni operator mutacije kako bi se pojačala sposobnost lokalnog pretraživanja u prostoru odlučivanja koji je mapiran regionom kolena i da bi se izbeglo zaglavljivanje u lokalnom optimumu regiona kolena. Koristi se Cauchy-jeva mutacija, koja se može predstaviti sledećim formulama:

$$x_i(t+1) = x_i(t) + \xi * C(\mu, \sigma) \tag{3.18}$$

$$g(x) = \frac{a}{\pi * (x^2 + a^2)} \tag{3.19}$$

gde je $C(\mu, \sigma)$ Cauchy-jeva slučajna funkcija sa srednjom vrednošću μ i varijansom σ, a $g(x)$ funkcija gustine Cauchy-jeve raspodele.

Modifikovani višeciljni brain storm algoritam koji primenjuje strategiju klasterovanja u prostoru ciljeva, DBSSCAN algoritam za klasterovanje i DE strategiju za mutaciju je dat u [86]. Strategija klasterovanja u prostoru ciljeva radi tako da sugeriše potencijalne Pareto-dominantne oblasti u sledećoj iteraciji, a u ovom rešenju se koristi algoritam za pronalaženje klastera u velikim prostornim datotekama sa šumom, baziran na gustini [87]. Predloženi algoritam polazi od činjenice da je glavni razlog na osnovu kojeg mi prepoznajemo klaster (tačaka) to što je u okviru klastera veća gustina tačaka nego van klastera, npr. u oblasti šuma. Ključna ideja je da za svaku tačku

klastera, okruženje koje se nalazi u opsegu nekog radijusa te tačke mora da sadrži barem minimalni broj tačaka, tj. gustina u okruženju mora da premaši neku graničnu vrednost. Proces generisanja nove individue se sastoji iz dva operatora: mutacije i selekcije. Mutacija se bazira na strategiji diferencijalne mutacije zato što su Gauss-ov i Cauchy-jev operator mutacije pokazali loše performanse u vidu spore konvergencije kada su primenjivani za rešavanje višeciljnih i multimodalnih optimizacionih problema. Kada se treba generisati nova ideja na osnovu tekućih, mora se uzeti u obzir razlika između tekućih ideja. Ako sa X_{new} obeležimo novo generisanu ideju, sa $X_{selected}$ tekuću ideju, pri čemu su $X_a = (x_a^1, x_a^2, \ldots, x_a^d)$ i $X_b = (x_b^1, x_b^2, \ldots, x_b^d)$ dve različite slučajne ideje odabrane da predstavljaju razliku ideja, onda se postupak generisanje nove ideje primenom diferencijalne mutacije može predstaviti sledećom jednačinom:

$$x_{new}^d = x_{selected}^d + rand(0,1)_d * (x_a^d - x_b^d) \tag{3.20}$$

Operator selekcije se bazira na Pareto dominaciji. Pareto skup se ažurira novim nedominiranim rešenjima. U ovom koraku, svako novo nedominirano rešenje dobijeno u tekućoj iteraciji se poredi sa svim ostalim članovima Pareto skupa. Ako veličina Pareto skupa prelazi maksimalnu vrednost, vrši se odsecanje uz uzimanje u obzir diverziteta.

Za razliku od jednociljnih optimizacionih problema, multimodalna optimizacija zahteva da se istovremeno nađe više lokalnih i globalnih optimuma. U [88] se za rešavanje multimodalnih problema predlaže samoadaptivni brain storm optimizacioni algoritam (Self-adaptive Brain storm Optimization, SBSO). Ovaj algoritam koristi max-fitnes grupisanje kao metod klasterovanja, koji se može predstaviti pseudokodom u Algoritmu 3.14.

Algoritam 3.14: Max-fitnes grupisanje kao metod klasterovanja

```
1 begin
2     Na slučajan način u prostoru pretraživanja inicijalizovati populaciju P
        koja ima N ideja;
3     Naći ideju koja ima najbolju fitnes vrednost i označiti je kao seme X;
4     Kombinovati M − 1 ideja iz populacije P koje su najbliže ideji X da bi se
        formirala podpopulacija;
5     Eliminisati ovih M ideja iz P;
6     Ponavljati korake 2-4 dok se populacija P ne podeli u P/M podpopulacija
7 end
```

Da bi se uskladili procesi eksploracije i eksploatacije, primenjuje se samoadaptivni kontrolni parametar. Primenom ovog parametra algoritam brzo konvergira ka različitim optimumima u različitim klasterima. Operator mutacije funkcioniše tako da se nova ideja $u_{i,j}$ generiše primenom binomne krosover operacije na ideji iz poslednje iteracije $x_{i,j}$ i ideji posle mutacije $v_{i,j}$:

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{ako} \quad rand_j(0,1) \leq C_r \quad \text{ili} \quad j = j_{rand} \\ x_{i,j}, & \text{inače} \end{cases} \tag{3.21}$$

gde je $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, n_d$, j_{rand} je slučajna celobrojna vrednost iz $\{1, 2, ..., n_d\}$, a $rand_j(0,1)$ je slučajna vrednost iz intervala (0,1). U svakoj generaciji, krosover vrednost Cr_i svake ideje se nezavisno generiše u skladu sa normalnom distribucijom srednje vrednosti Cr_m i standardne devijacije 0.1.

$$Cr_i = rand(Cr_m, 0.1) \tag{3.22}$$

$$Cr_m = mean(S_{cr}) \tag{3.23}$$

gde je S_{cr} skup svih uspešnih krosover vrednosti iz prethodne generacije. Na početku Cr_m je 0.5, a S_{cr} je prazan skup. Operacija selekcije se obavlja poređenjem (fitnes vrednosti) u_i i najbliže individue u populaciji, x_s.

3.4 Teorijska analiza

Kao što je rečeno, u originalnom BSO algoritmu je moguće identifikovati tri glavna operatora: operator grupisanja, operator zamene i operator kreiranja. U ovim operatorima pojavljuju se tri kontrolna parametra, odnosno tri verovatnoće i to: (1) p_{5a} koji reguliše operator zamene, tj. kontroliše da li će centar klastera biti zamenjen nekom slučajno generisanom idejom (narušavanje centra klastera); (2) p_{6b} koji reguliše operator kreiranja, tj. kontroliše da li će se nove ideje kreirati na osnovu jednog ili dva klastera; i (3) p_{6bii} i p_{6c} koji kontrolišu da li će se prilikom kreiranja novih ideja koristiti centar klastera ili neka slučajno odabrana ideja.

Ispitivanje kako ovi parametri utiču na performanse BSO i MBSO algoritama je urađeno u [89]. Rezultati simulacija pokazuju da p_{5a} vrednost ima veoma mali uti-

caj na performanse algoritma. Potrebna su nova istraživanja mogućnosti modifikacije operatora zamene, a uočava se i da je moguće napraviti BSO varijante bez primene operatora zamene koje neće degradirati performanse algoritma. Kada je reč o parametru p_{6b}, pokazuje se da on ima veći uticaj na BSO nego na MBSO, pri čemu je velika vrednost p_{6b} dobra za MBSO, a mala vrednost parametra p_{6b} je bolje rešenje za BSO. Kada je vrednost p_{6b} manja, BSO ima veću šansu da generiše nove ideje na osnovu dva klastera, što pruža mogućnost algoritmu da iskoristi više informacija iz populacije i na taj način poboljša performanse. S druge strane, vrednosti parametara p_{6bii} i p_{6c} imaju veći uticaj na MBSO nego na BSO. U slučaju BSO algoritma, veće vrednosti paramentra p_{6bii} daju bolje rezultate kod primene na unimodalnim funkcijama; suprotno važi u slučaju primene na multimodalnim funkcijama. Razlog može biti to što je veća verovatnoća da BSO koristi centar klastera za kreiranje novih ideja ako je p_{6bii} veće. To može da poveća brzinu konvergencije i poboljša performanse u slučaju primene na unimodalnim benčmark funkcijama. Kada je vrednost p_{6bii} mala, BSO koristi informacije iz populacije za kreiranje novih ideja i stoga je to dobro rešenje za multimodalne funkcije. Međutim, ako je $p_{6bii} = 0$, to može da ima štetne posledice na performanse BSO algoritma. Kada je reč o MBSO, pokazuje se da ni male ni velike vrednosti p_{6bii} ne daju dobre rezultate. U ovom slučaju najbolji rezultati u pogledu ostvarivanja balansa između eksploracije i eksploatacije se dobijaju ako p_{6bii} ima srednju vrednost od oko 0.4.

Problem preuranjene konvergencije je tipičan za sve algoritme inteligencije rojeva. Dešava se kada se rešenja grupišu u klastere (konvergiraju), a zatim ne dolazi do njihove divergencije. Mehanizam klasterovanja koji se koristi u BSO algoritmu nema štetne efekte već služi usmeravanju individua ka boljim oblastima rešenja. Analiza strategija klasterovanja i ostalih osobina BSO algoritma je obavljena u [90]. Analizom na tri unimodalne i tri multimodalne funkcije je utvrđeno da BSO algoritam ima brzu konvergenciju u početku pretraživanja, što pokazuje da dovoljno dobri regioni mogu biti locirani nakon nekoliko klasterovanja. Međutim, potrebno je poboljšati prevenciju preuranjene konvergencije, kako bi sposobnost izvlačenja algoritma iz lokalnog optimuma bila bolja. Uopšteno gledajući, algoritam ima bolje performanse prilikom rešavanja unimodalnih tj. jednostavnih problema, dok kod rešavanja složenih funkcija ima mnogo oscilatornog kretanja.

Postoji veliki broj različitih optimizacionih problema: jednociljni, višeciljni, ograničeni, kombinatorni, multimodalni, te algoritmi za rad u fiksnom i dinamičkom okruženju. Poznato je da ne postoji jedan algoritam koji najbolje rešava različite optimizacione probleme. Veoma je teško naći algoritam koji će biti najbolji za rešavanje neke vrste problema ako prethodno nemamo znanje o problemu i njegovom okruženju. Stoga se postavlja pitanje da li i kako optimizacioni algoritam može da razvije svoj kapacitet učenja da bi bolje rešio probleme koji nisu poznati u vreme kreiranja algoritma. Dakle, idelan optimizacioni algoritam treba da ima sposobonost samoadaptacije kako bi stekao kapacitet za učenje i rešavanje problema u njegovom specifičnom okruženju, tj. da razvije potencijal i kapacitet učenja koji odgovara problemu i njegovom okruženju, što će onda omogućiti algoritmu da bolje uči i da problem efikasno reši. Neophodnost ugrađivanja razvojnog učenja u algoritme inteligencije rojeva i analiza nekoliko algoritama inteligencija rojeva iz perspektive razvojnog učenja su obavljeni u [35]. Razvijen je okvir razvojnog algoritma inteligencije rojeva (developmetal swarm intelligence algorithm) koji treba da pomogne razumevanju postojećih i implementaciji novih razvojnih algoritama inteligencije rojeva i razvojnih evolutivnih algoritama.

3.5 Primene BSO algoritma

BSO algoritam se pokazao uspešnim u rešavanju realnih problema u različitim oblastima kao što su: elektromagnetika i energetika, komunikacije, bežične mreže, aeronautika, finansije i sl. U ovim primenama se obično koriste posebne modifikacije originalnog BSO pogodne za rešavanje specifičnog problema u datoj oblasti. U elektroenergetskim sistemima postoji problem minimizacije troškova prilikom proizvodnje zahtevane količine električne energije (economic dispatch problem, ED). Ovo je optimizacioni problem koji posebno dolazi do izražaja u sistemima koji koriste energiju vetra jer ju je teško predvideti. U [91] se primenjuje BSO algoritam na rešavanje ED problema u sistemu koji ima termoelektrane i vetroelektrane. Hibridni algoritam koji se zasniva na brainstorming procesu i optimizaciji baziranoj na podučavanju i učenju (Teaching–Learning–Based Optimization, TLBO) se predlaže za rešavanje ED problema u [92]. Nalaženje optimalne lokacije i podešavanje fleksibilnih AC sistema za prenos energije je složen višeciljni, multimodalni optimizacioni problem sa ograničenjima koji se u [93] pokušava rešiti primenom BSO algoritma. Brain storm

optimizacija se koristi za rešavanje ED problema i u [94], [95], [96].

Raspoređivanje poslova je važno pitanje u sistemima gde je potrebna minimizacija vremena izvršavanja. Optimizacija troškova i minimizacija vremena izvršavanja su međusobno konflikni ciljevi zato što su brži resursi obično i skuplji. Višeciljni algoritam baziran na BSO algoritmu koji rešava optimizacioni problem raspoređivanja poslova u grid okruženju je prikazan u [97].

Performanse bežične senzorske mreže u mnogome zavise od strategije raspoređivanja čvorova u mreži. Međutim, nalaženje optimalne strategije raspoređivanja čvorova u bežičnoj senzorskoj mreži koja bi ispunila više ciljeva kao što su smanjenje troškova, robustnost na otkaz čvorova, smanjenje vremena izračunavanja i garantovanje visoke pokrivenosti uz očuvanje povezanosti je veoma težak optimizacioni problem. Jedno od mogućih rešenja ovog problema uz pomoć modifikovanog BSO algoritma je dato u [76].

Predviđanje berzanskih indeksa je važan i neophodan alat kako za investitore tako i za vladu. Međutim, zbog velike promenljivosti, visokog šuma i nelinearnosti berzanskih indeksa, ovo predviđanje je zahtevan zadatak i složen optimizacioni problem. U [98] se on pokušava rešiti primenom hibridnog pristupa kombinujući BSO algoritam i model gray neuralne mreže (gray neural networks, GNN) tako što se inicijalizacija parametara gray neuralne mreže vrši uz pomoć BSO. Pokazuje se da predloženi algoritam ima sposobnost rešavanja nedostataka tradicionalnog GNN modela sa slučajno inicijalizovanim parametrima putem rešavanja problema lokalnog optimuma i male tačnosti predviđanja.

Optimalna rekonfiguracija formacije koju čini više satelita u geostacionarnoj orbiti je optimizacioni problem sa ograničenjima kao što su: minimalna potrošnja goriva, finalna konfiguracija i izbegavanje kolizija. Za rešavanje ovog problema Sun i dr. [99] predlažu modifikovani BSO algoritam baziran za zatvorenoj petlji (close-loop BSO). Razvijene su tri verzije ovog algoritma, koji zamenjuje operator kreiranja iz originalnog BSO strategijom zatvorene petlje. Ovaj pristup poboljšava performanse tako što koristi povratne informacije iz procesa pretraživanja.

BSO algoritam je našao primenu i u aeronautici, za dizajn automatskih sistema za sletanje aviona na nosač aviona. Ovaj sistem je veoma kritičan zato što mora da omogući sletanje u izuzetno teškim uslovima kao što su loša vidljivost, jak vetak

i nepovoljni uslovi na moru. Novi metod za optimizaciju kontrolnih parametara u automatskom sistemu za sletanje na nosač aviona za F/A-18A, baziran na pojednostavljenom BSO algoritmu je razvijen u [100].

Na kraju, pomenimo i primenu BSO algoritma u elektromagnetici: za rešavanje problema Loney-evog solenoida [101] i za DC motore bez četkica [102].

4 PLANIRANJE PUTANJE ROBOTA

4.1 Osnovni pojmovi

Ubrzani razvoj tehnologije kreira nove i poboljšane mogućnosti u mnogim različitim aspektima života. U današnje vreme, jedna od tehnologija koja obećava je tehnologija robota. Robotika je multidisciplinarna naučna oblast koja uključuje računarstvo, elektrotehniku, mehaniku i mašinstvo, i mnoge druge, zato što se bavi dizajnom robota, njihovom konstrukcijom, te razvojem računarskih sistema za njihovu kontrolu, za obradu podataka i td.

Robot je programabilna mašina koja imitira akcije ili ima izgled inteligentnog bića, najčešće čoveka, i mora imati sledeće osobine: 1) mogućnost percepcije sredine (sensing and perception) kako bi dobio informacije iz okruženja; 2) sposobnost za izvršavanje različitih zadataka: pokretljivost ili izvršavanje fizičkih radnji kao što su pomeranje ili manipulacija objektima i sl; 3) reprogramiranje; 4) autonomno funkcionisanje i/ili interakcija sa ljudskim bićima. Institut za robote iz Sjedinjenih Američkih Država (Robot Institute of Amerika) je 1979. godine definisao robot kao reprogramabilni, multifunkcionalni manipulator koji je putem različitih programiranih pokreta napravljen da pomera materijal, delove, alate ili specijalizovane uređaje radi obavljanja različitih zadataka. Roboti imaju mogućnost da obavljaju zadatke bez pomoći čoveka ili uz malu ljudsku angažovanost, i obično obavljaju zadatke koji su opasni, stresni, naporni ili dosadni za čoveka. Postoje različite vrste robota od kojih navodimo: robot manipulator, mobilni robot manipulator, robot sa nogama (legged robot), mobilni robot sa točkovima (wheeled mobile robot), podvodni robot ili autonomno podvodno vozilo (underwater robot, autonomous underwater vehicle), vazdušni robot ili bespilotna letelica (arial robot, unmanned aerial vehicle), humanoidni robot. Osnovne komponente robota su: baza, mikrokontroler, korisnički interfejs, senzori, aktuatori, jedinica za konverziju energije i manipulatori. Manipulatori se prave povezivanjem čvrstih tela, koji se nazivaju delovi (links) pomoću spojeva (joints), tako da je omogućeno relativno pomeranje, odnosno, pokretanje susednih delova. Pokretanje (aktuacija) spojeva se obavlja elektromehanički, obično pomoću električnih motora, što daje mogućnost robotu da izvrši određeni fizički zadatak [103]. Baza robota može biti fiksna, kao kod robot manipulatora koji se koriste u industriji, ili mobilna, koja se pravi kao platforma

sa pokretnim delovima poput nogu ili točkova. Razvoj nauke i tehnologije omogućio je pojavu robota u svakodnevnom životu, a posebno brzo raste broj robota u industrijskom i uslužnom sektoru. Najčešće primene robota u industriji uključuju: rukovanje i transfer materijala, spajanje i razdvajanje delova, sklapanje mašina i komponenata, zavarivanje, farbanje, inspekcija i sl.

Bespilotne letelice (unmanned aerial vehicles, UAV) predstavljaju daljinski upravljive ili samoupravljive letelice. Postale su poznate i pod imenom „robotske letelice“ i njihova upotreba je široko raspostranjena, kako u vojne tako i u civilne svrhe. One imaju brojne prednosti kao što su niska cena, dobra sposobnost manevrisanja i visoka stopa preživljavanja. UAV su pogodnije za nadgledanje, izviđanje, osmatranje u složenim i opasnim okruženjima od vozila kojima upravlja čovek. Uz navedene kvalitete, UAV letelice imaju mogućnost da sadrže i različite uređaje i opremu poput kamera, senzora, oružja i sl. Inicijalno su ove letelice korišćene u vojne svrhe i bile su poznate pod nazivom borbene bespilotne letelice (unmanned combat aerial vehicles, UCAV). Međutim, u sadašnje vreme, zbog navedenih kvaliteta i osobina, UAV se primenjuju u različitim oblastima kao što su poljoprivreda, industrija, nadgledanje i osmatranje, ali imaju i mnoge druge naučne i komercijalne primene. Više faktora je doprinelo razvoju upotrebe UAV letelica: 1) tehnološki razvoj je omogućio pojavu moćnih senzora, mikroprocesora i drugih sistema kojima se ostvaruje efikasnost i autonomnost koja premašuje ljudske mogućnosti; 2) uspešnost upotrebe ovih letelica u vojne svrhe je doprinela povećanju ulaganja u razvoj tehnologija za UAV; 3) UAV mogu da rade u okruženjima i pod uslovima gde vozila kojima upravljaju ljudi ne mogu, kao što su veoma velike ili veoma male nadmorske visine. Tehnologije koje omogućuju UAV letelicama autonomno ponašanje i operabilnost bez prisustva čoveka su: navigacioni senzori i mikroprocesori; sofisticirani komunikacioni sistemi; i zemljani off-board sistemi za komandovanje, komunikaciju i kontrolu (C3). Kada je u pitanju C3 infrastruktura, mogu se identifikovati sledeća pitanja od interesa: interfejs čovek-mašina, kontrola glasa, C3 sistemi za više letelica (multiaircraft C3), identifikacija mete itd. UAV letelice se na osnovu njihovih karakteristika mogu klasifikovati u sledeće četiri grupe [104]:

- UAV sa fiksnim krilima (fixed-wing UAV) koji se još nazivaju i avioni bez pilota (unmanned airplanes). Oni zahtevaju zalet da bi uzleteli ili lansiranje iz

katapulta;

- UAV sa rotirajućim krilima (rotary-wing UAV, rotorcraft UAV) imaju vertikalno uzletanje i sletanje što im daje visoku mogućnost manevrisanja. Moguće su različite konfiguracije ovih letelica: sa glavnim i sporednim propelerima (kao kod helikoptera), koaksijalnim propelerima, tandem propelerima, višestrukim propelerima itd;

- Letelice u obliku balona i vazdušnih brodova, koje su obično velike, imaju dobru izdržljivost, i kreću se manjim brzinama;

- UAV sa mašućim krilima (flapping-wings UAV), čija je konfiguracija inspirisana krilima insekata.

Osim navedenih postoje i hibridne konfugracije, npr. vertikalno poletanje kao helikopter, ali let kao avion.

Vozilo je objekat koji ima mogućnost kretanja i može se shvatiti kao robot koji ne sadrži manipulatore. Za predstavljanje vozila služe vektori pozicije i orjentacije, i geometrijski model. Vektor pozicije definiše položaj vozila u dvodimenzionalnom ili trodimenzionalnom prostoru. Prostor sveta (world space) je fizički prostor u kome se vozilo kreće, i obično je to trodimenzionalni euklidski prostor, mada se nekad zbog pojednostavljivanja problema može posmatrati i kretanje u dvodimenzionalnom prostoru. Konfiguracija vozila je skup vrednosti koje jedinstveno određuju vozilo i obično sadrži šest vrednosti: tri komponente vektora pozicije i tri komponente vektora orjentacije. Ukoliko robot sadrži manipulatore konfiguracija je složenija zato što stepen slobode svakog manipulatora dodaje po jedan parametar u konfiguraciju. Konfiguracioni prostor (configuration space, C-space) čini skup svih mogućih konfiguracija koje vozilo može da ima. U analizi kretanja robota često je potrebno da se uz konfiguraciju uključi i pojam stanja, koje se sastoji od konfiguracije robota i parametara vezanih za promenu konfiguracije. U slučaju letelice, stanje se sastoji od 12 parametara: tri koordinate pozicije, tri koordinate brzine, tri orjentaciona ugla i tri ugla promene orjentacije.

Skup svih mogućih stanja se naziva prostor stanja (state space). Broj parametara koji je potreban za predstavljanje konfiguracije ili stanja se naziva broj stepena slobode (number of degrees of freedom). Prostor sveta, odnosno, prostor konfiguracija ili stanja

se može podeliti u dva prostora: slobodni prostor (free space) koji predstavlja skup tačaka koje vozilo može zauzimati tokom kretanja, i prostor prepreka (obstacle space), koji predstavlja skup tačaka gde vozilo ne sme da se nađe, jer bi tada bilo u koliziji sa drugim objektima. Putanja je kriva koju vozilo opisuje tokom kretanja. Ona ne mora biti glatka, ne uzima u obzir vreme kao parametar, a može biti sastavljena od segmenata tako da svaki segment bude trajektorija. Trajektorija je putanja koja uključuje i vreme kao parametar i može se matematički opisati kao polinomska funkcija vremena $X(t)$, tako da brzina i ubrzanje mogu da se izračunaju određivanjem izvoda ove funkcije. Ona mora da uvaži kinodinamička ograničenja, kao što su ograničenja brzine, ubrzanja, okretanja, promenu pravca i sl.

Planiranje kretanja (motion planning) može da se definiše kao planiranje putanje ili trajektorije, i ima za rezultat putanju ili trajektoriju od početnog stanja ili konfiguracije do ciljnog stanja ili konfiguracije [105]. Planiranje putanje ima za cilj nalaženje kontinualne krive u konfiguracionom prostoru koja počinje od startne pozicije, a završava se u krajnjoj poziciji. Planiranje trajektorije obično nastupa nakon planiranja putanje tako što uzima rešenje algoritma za planiranje putanje i određuje kako robot može da se kreće po datoj putanji. Algoritam za planiranje putanje je kompletan ako je uspešan u generisanju putanje ukoliko ta putanja postoji, odnosno, prijavljuje da putanju nije moguće generisati ukoliko putanja zaista ne postoji. Dobar planer (sound planner) putanje je onaj koji uvek garantuje da će robot dostići željenu destinaciju uz obavezno izbegavanje sudara sa preprekama. Ovo je ujedno i jedna od najvažnijih osobina planera, posebno za UAV vozila, jer posledice kolizija mogu da uzrokuju trajna oštećenja i katastrofalne posledice.

Problemi planiranja kretanja se mogu podeliti na više načina. Problem se naziva statičkim ukoliko je poznato kompletno znanje o okruženju. Ako znanje o okruženju nije kompletno ili se menja u toku kretanja robota, problem je dinamički. Kada se prepreke ne kreću, problem je vremenski nepromenljiiv (time-invariant), u suprotnom, govori se o vremenski promenljivom problemu (time-variant). Diferencijalno ograničen problem je onaj gde jednačine kretanja vozila predstavljaju ograničenja, tako da putanja mora biti trajektorija dinamičkog sistema. U najpoznatije tipove problema planiranja kretanja spadaju: tačkasti robot (point robot), tačkasti robot sa diferencijalnim ograničenjima, Jogger-ov problem, problem bube (bug problem), problem težinskog

regiona (weighted region problem), Mover problem, opšte vozilo sa diferencijalnim ograničenjima, vremenski promenljiva okruženja i multimover problem.

4.2 Problem planiranja putanje

Neka je dat mobilni robot sa k stepena slobode koji može da se kreće u dvodimenzionalnom ili trodimenzionalnom prostoru. Prostor sadrži prepreke koje su poznate robotu. Ako pretpostavimo da su inicijalna i željena krajnja pozicija robota poznate, problem kretanja robota je: 1) određivanje da li je kontinualno kretanje robota od inicijalne do željene pozicije moguće; 2) planiranje tog kretanja ukoliko je odgovor pod 1) pozitivan. Osnovna pitanja od interesa, odnosno koraci u formulaciji problema planiranja kretanja su sledeći: izračunavanje konfiguracije, reprezentacija objekta, pristupi u planiranju kretanja, metode pretraživanja i lokalna optimizacija kretanja [106].

Planiranje putanje mobilnih robota je veoma aktivna oblast naučnog istraživanja još od 60-tih godina prošlog veka. Veću pažnju istraživača ova oblast dobija nakon objavljivanja rada [107]. U njemu se predlaže algoritam za planiranje putanje bez kolizija za slučaj kretanja poliedarskog objekta među preprekama koje su istog oblika. Algoritam radi na iterativan način, a počinje tako što se formira putanja u obliku prave linije između početne i krajnje tačke. Ukoliko takva putanja sadrži mesta kolizija, na osnovu informacija o koliziji predlaže se nova putanja, i dati postupak se ponavlja sve dok putanja ne sadrži kolizije.

U zavisnosti od toga da li je robotu dostupno kompletno znanje o okruženju u kome se kreće, planiranje putanje se može podeliti u dve vrste: globalno planiranje putanje ili deliberativni pristup, i lokalno planiranje putanje ili reaktivni pristup. Globalno planiranje putanje (global path planning, GPP) je postupak kojim se određuje putanja bez prepreka u slučaju kada robot ima kompletne informacije o okruženju. GPP se može izvršiti van mreže (offline). Ako su informacije o okruženju samo delimično poznate ili su potpuno nepoznate, govori se o lokalnom planiranju putanje (local path planning, LPP), koje se još naziva i online planiranje putanje. U slučaju kada je teren nedovoljno poznat ili neodređen, GPP postupak nije dovoljno robustan i tada se za nalaženje optimalne putanje može uspešno koristiti LPP.

4.2.1 Ograničenja putanje

Prilikom planiranja putanje ili trajektorije moraju se uzeti u obzir brojni zahtevi i ograničenja. Putanja kojom se robot kreće treba da bude najkraća moguća tako da potrošnja goriva bude minimalna, kao i da se od startne do ciljne pozicije stigne za najkraće vreme. Osim toga, zahteva se da ne dolazi do kolizije robota sa preprekama ili sa drugim robotima, a u slučaju UAV letelica potrebno je minimizovati izlaganje UAV pretnjama. Letelica mora da leti po putanji tako da je verovatnoća izlaganja pretnjama u vidu neprijateljskih radara, raketa ili drugih letelica minimalna. Ograničenja mogu da se odnose na kinodinamička svojstva robota ili ograničenja sredine. Dodatna ograničenja UAV letelice koja moraju biti razmatrana prilikom planiranja putanje su: ograničenja ugla okretanja (turning angle) α, ograničenja ugla uspona/poniranja (climbing/diving angle) β, visina leta h i ugao prilaska ciljnoj poziciji. Ograničenje ugla okretanja zahteva da putanja bude takva da je okretanje letelice manje ili jednako određenoj graničnoj vrednosti kako ne bi došlo od oštećenja u strukturi letelice ili sudara sa drugim letelicama. Ograničenje ugla uspona/poniranja ima isti smisao kao prethodno ograničenje, ali samo u smeru altitude. Može biti pozitivno ili negativno, što zavisi od smera kretanja (npr. negativno je ako je u pitanju poniranje). Zahteva se da nagle promene budu izbegnute kako bi se minimizovao rizik od kolizija. Ograničenje minimalne visine leta je potrebno zbog smanjenja verovatnoće detekcije od strane neprijatelja, jer nizak let omogućuje maskirajući efekat terena. S druge strane, veoma mala visina leta nosi rizik rušenja letelice i trajnog oštećenja, što se mora izbeći. U nekim primenama, kao što su operacije napada, unapred je određen optimalni smer kretanja, tako da robot mora da sledi taj smer, odnosno ugao prilaska ciljnoj poziciji.

Osim ovoga, moraju se razmotriti i osobine i ograničenja sredine, kao što su rastojanja između prepreka, verovatnoća pojavljivanja dinamičkih prepreka, konfiguracija terena, zabranjene zone leta, zone pretnji, opseg mape leta i sl. Okruženje u kome se robot kreće može da ima različite karakteristike u pogledu prepreka i pretnji sa kojima se robot suočava. Prepreke mogu da budu statičke ili pokretne, a izvori pretnji mogu da budi stalno prisutni i izvesni ili neizvesni, gde pod neizvenošću podrazumevamo da je nemoguće ili previše skupo unapred odrediti njihovu preciznu poziciju, kao što je npr. vatra u misijama spašavanja ili neprijatelji i mine na bojnom polju.

Diskontinuiteti na krivoj koja predstavlja putanju nose brojne nedostatke kao što su nestabilnost kontrolnog sistema, prekoračenje željene destinacije (overshooting), neprijatan osećaj putnika, a u nekim primenama mogu da izazovu mehaničko oštećenje i kvar. Pokazalo se da putanja koja ima oblik kontinualne krive poboljšava stabilnost i kontrolu industrijskih vozila [108]. Kinodinamička ograničenja robota zahtevaju konstruisanje kontinualne glatke krive putanje pri čemu su ti zahtevi još važniji za UAV letelice zato što su njihova kinodinamička ograničenja znatno strožija. Naime, letelice ne mogu da se kreću po isprekidanim linijskim putanjama zato što nagli zaokreti mogu da budu štetni za strukturu letelice. Metode i strategije za obezbeđenje glatkosti (path smoothing) su veoma često deo planera putanje. Nekada su inkorporirane u optimizacioni algoritam, a nekada se realizuju nekon generisanja optimalnih putanja u obliku pravih linija.

Glatkost se može predstaviti kao suma uglova skretanja formiranih od strane bilo koja tri susedna čvora na putanji. Direktno računanje glatkosti je obično vremenski veoma zahtevno, te se često primenjuju postupci za indirektno izračunavanje. Jedan od mogućih je dat u [109] i on koristi dva parametra, S_c i S_p, da bi se izračunala glatkost putanje. Parametar S_c je odnos broja uglova skretanja (deflection angles) koji su manji od date očekivane vrednosti i ukupnog broja uglova skretanja, a S_p je odnos broja segmenata putanje koji su veći od broja segmenata u putanji koja ima najmanji broj segmenata i ukupnog broja segmenata putanje. Glatkost se tada može izračunati na sledeći način:

$$S = \alpha * S_c + \beta * S_p \tag{4.1}$$

$$S_c = 1 - \frac{DA_l}{N_f - 1} \tag{4.2}$$

$$S_p = 1 - \frac{S_{min}}{N_f} \tag{4.3}$$

pri čemu je N_f ukupan broj segmenata putanje, DA_l je broj uglova skretanja većih od očekivane vrednosti; S_{min} je broj segmenata u putanji sa najmanjim brojem segmenata; α i β su težinski koeficijenti.

Za robote koji imaju ograničenje ugla okretanja često su se koristile Dubin-ove putanje kao rešenje kojim se postiže glatkost. Dubinova istraživanja [110] su pokazala da se za robote sa poznatom inicijalnom i krajnjom konfiguracijom u dvodimenzio-

nalnom okruženju bez prepreka može konstruisati najkraća putanja takva da pripada nekoj od šest mogućih tipova putanja. Ona mora da se sastoji od najviše tri dela koja su u obliku pravih linija ili kružnih lukova. Rešenja koja se baziraju na ovom su predlagana i za postizanje glatkosti UAV letelica u 2D i 3D okruženju [111], [112].

Za postizanje glatkosti putanje u poslednje vreme se često koriste algoritmi bazirani na splajn krivama među kojima su najpoznatije Bezier-ove krive, B-splajn krive i NURBS (NonUniform Rational B-Splines) krive [113]. Bezier-ove krive su korišćene u CAD aplikacijama i razvijene su za potrebe automobilske industrije. Bezier-ova kriva $c(u)$ n-tog stepena je definisana jednačinom (4.4), gde je u normalizovani parametar krive, a $B_{n,i}(u)$ je Bezier-va blending funkcija za i-tu kontrolnu tačku P_i . Ove funkcije su prikazane jednačinom (4.5).

$$c(u) = \sum_{i=0}^{n} B_{n,i}(u) P_i \tag{4.4}$$

$$B_{n,i}(u) = \frac{n!}{i!\,(n-i)!} u^i (1-u)^{n-i} \tag{4.5}$$

Blending funkcije nemaju lokalni uticaj na generisanu krivu, što može da bude ograničavajući faktor u situacijama kada se zbog detektovanih prepreka zahteva modifikacija putanje. Primenom Bezier-ovih krivih putanja se može predstaviti manjim brojem parametara u odnosu na slučaj kada se koristi kompletan geometrijski opis putanje, što znatno poboljšava performanse celog algoritma. Zbog toga se Bezier-ove krive sa različitim brojem kontrolnih tačaka koriste za postizanje glatkosti putanje. Red Bezier-ove krive zavisi od broja kontrolnih tačaka, i konstrukcija Bezier-ovih krivih zahteva da se unapred definišu sve koordinate (horizontalne i vertikalne) odgovarajućih kontrolnih tačaka jer je u suprotnom za generisanje putanje neophodno korišćenje krivih višeg stepena, što je u slučaju dugačkih putanja računarski neefikasno.

B-Spline krive se uglavnom koriste za generisanje putanje industrijskih robota manipulatora, a njihova upotreba za mobilne robote je još uvek ograničena. B-Spline kriva p-tog stepena, $c(u)$, se definiše pomoću n kontrolnih tačaka P_i i vektora čvorova $\hat{u}$ koji sadrži m neopadajućih realnih brojeva, pri čemu je broj čvorova m jednak $n+p+1$. Ako je u normalizovani parametar dužine krive, a $N_{i,p}(u)$ i-ta bazna funkcija B-Spline

krive stepena p, ona se može predstaviti sledećom jednačinom:

$$c(u) = \sum_{i=0}^{n} N_{i,p}(u) P_i \tag{4.6}$$

Broj baznih funkcija je jednak broju kontrolnih tačaka, a za njihovo izračunavanje se koristi rekurzivni algoritam. Za razliku od Bezier-ovih krivih, red B-Spline krive ne zavisi od broja kontrolnih tačaka, i bazne funkcije imaju lokalni uticaj na generisanu krivu, što znači da je moguća modifikacija krive u situacijama kada je to neophodno, npr. za detekciju prepreke.

NURBS su težinska modifikacija neuniformnih B-splajn krivih (Non-Uniform B-splines) i njihova upotreba u robotici još uvek nije razvijena. Koriste se za primene gde se zahteva tačnost i računarska efikasnost, za generisanje putanje alata, modelovanje krvnih sudova, analizu konačnih elemenata i sl. NURBS krive su date jednačinom (4.7):

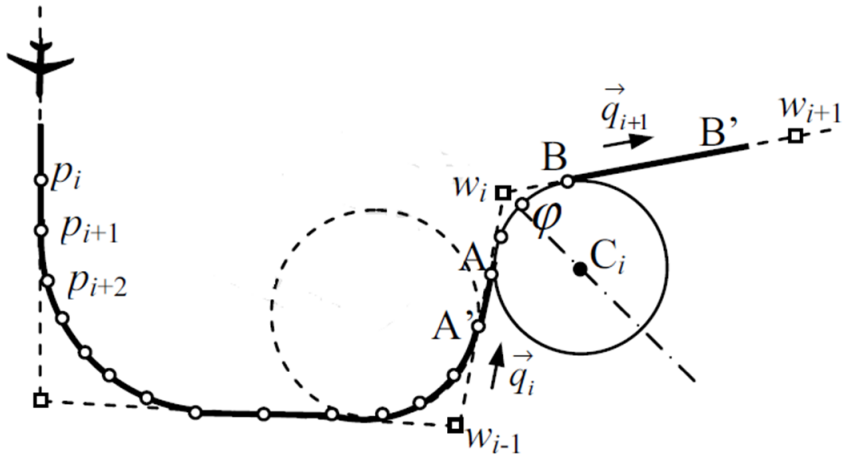
$$c(u) = \frac{\sum_{i=1}^{n} \omega_i N_{n,i}(u) P_i}{\sum_{i=1}^{n} \omega_i P_i} \tag{4.7}$$

Težinski koeficijenti ω_i su dodeljeni kontrolnim tačkama P_i, što omogućava da se kriva pomera prema definisanim kontrolnim tačkama. Uvođenje ovih koeficijenata daje veću fleksibilnost generisanju krivih menjanjem kontrolnih tačaka, čvorova i težina. Osobine NURBS krivih u pogledu baznih funkcija i reda krive su slične kao za B-Spline krive.

Jedan od načina za obezbeđenje glatkosti je primena dinamičke strategije k-trajektorija [112] koja se može opisati na sledeći način. Neka je dat segment putanje definisan trima kontrolnim tačkama ω_{i-1}, ω_i, ω_{i+1}, i neka je $\overrightarrow{q_i}$ jedinični vektor u smeru od ω_{i-1} do ω_i, a $\overrightarrow{q_{i+1}}$ jedinični vektor u smeru od ω_i do ω_{i+1}. Takođe, neka je φ ugao između $\overrightarrow{q_i}$ i $\overrightarrow{q_{i+1}}$, a $\hat{\mathrm{C}}$ kružni luk poluprečnika R i centra C_i koji leži na simetrali ugla formiranog od tri kontrolne tačke, pri čemu su R i C_i dati sledećim formulama:

$$R = 0.5 min\left\{\|\omega_i - \omega_{i-1}\|, \|\omega_{i+1} - \omega_i\|\right\} \tan\frac{\varphi}{2} \tag{4.8}$$

$$C_i = \omega_i + \left(\frac{R}{\tan\frac{\varphi}{2}}\right) \frac{\overrightarrow{q_{i+1}} - \overrightarrow{q_i}}{\|\overrightarrow{q_{i+1}} - \overrightarrow{q_i}\|} \tag{4.9}$$



Slika 4.1 Strategija glatkosti putanje

Kružni luk preseca linije ω_{i-1},ω_i i ω_i,ω_{i+1} pa se sada originalni segment putanje koji čine tri kontrolne tačke transformiše u segment definisan dvema linijama A'A i BB', i kružnim lukom $\hat{\text{C}}$, kao na Slici 4.1. Tačke putanje (waypoints) su prikazane kružićima, a kontrolne tačke kvadratićima.

Ovaj metod ima nekoliko prednosti: lako se integriše u algoritme za planiranje putanje koji generišu putanju u vidu pravih povezanih linija; metod ne unosi dodatne računarske zahteve zato što se glatke trajektorije generišu u realnom vremenu, u toku kretanja vozila po trajektoriji; minimizuje vreme odstupanja vozila od originalno generisane putanje u obliku pravih linija.

4.3 Metode rešavanja problema planiranja putanje

Planiranje putanje mobilnih robota je aktivna tema naučnog istraživanja od ogromne praktične važnosti, i mnogobrojne metode su predložene radi njegovog rešavanja. Generalno posmatrano, u mobilnoj robotici postoje dve vrste metoda za planiranje putanje: tradicionalne (klasične) i heurističke. Većina tradicionalnih metoda su prilikom rešavanja problema planiranja putanje robota uzimali u obzir samo minimizaciju dužine putanje. Tradicionalne metode zahtevaju veće vreme izračunavanja i imaju nedostatak u smislu zaustavljanja u lokalnom minimumu [114]. Kada se u naučnoj zajednici došlo do zaključka da je problem planiranja putanje NP-kompletan optimizacioni problem, pažnja je usmerena ka razvijanju i primeni heurističkih metoda za

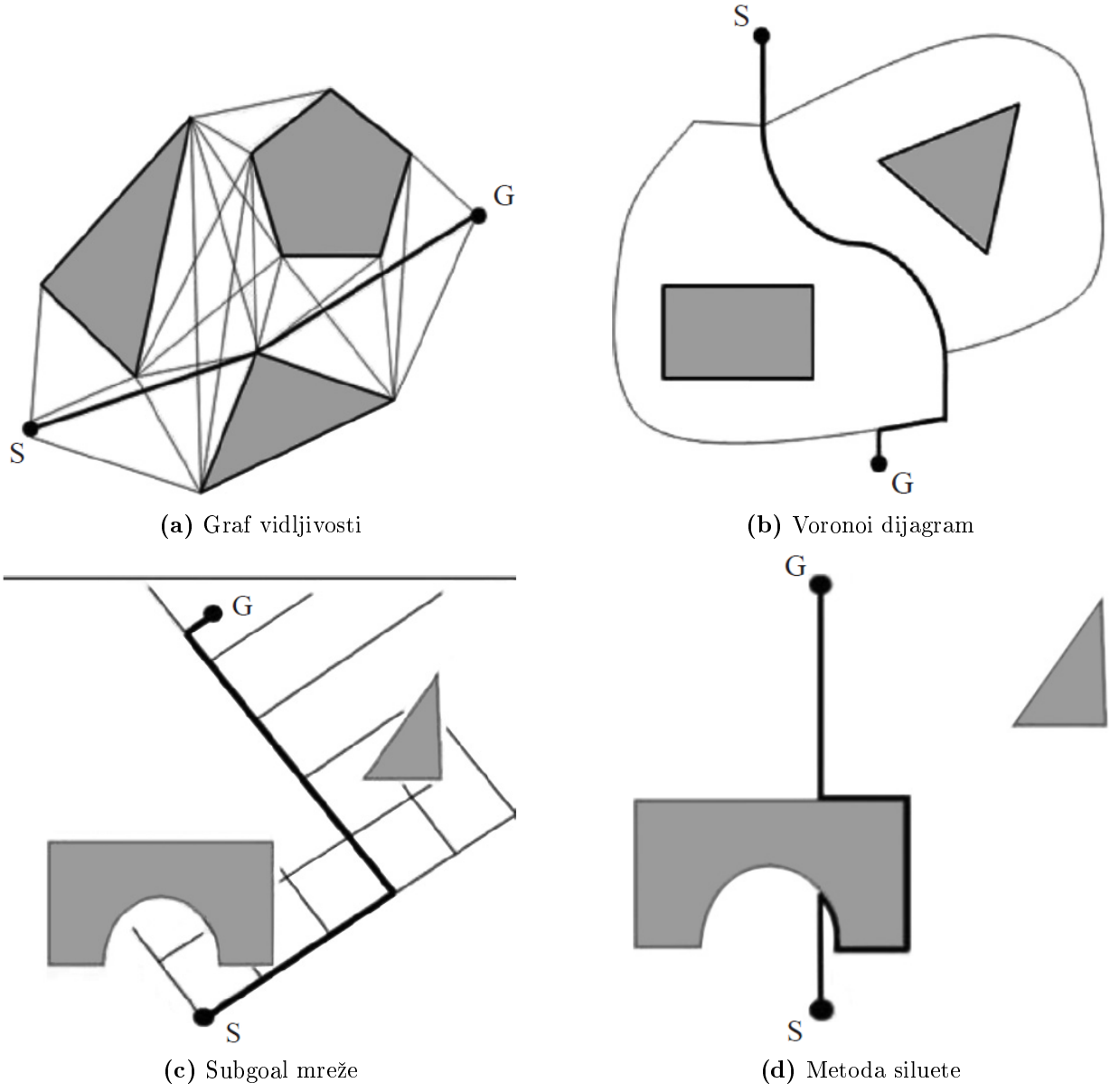
njegovo rešavanje.

Metode i pristupi za rešavanje problema planiranja putanje se mogu podeliti u nekoliko različitih grupa. Prvi pristup se naziva skeleton, koji je poznat i pod imenima roadmap ili autoput (highway) pristup. U ovom pristupu se konfiguracioni prostor mapira u mrežu jednodimenzionalnih linija i problem planiranja kretanja se tako transformiše u problem pretraživanja odgovarajućeg grafa. Planiranje kretanja se obavlja u tri koraka: 1) robot se pomera od startne konfiguracije do tačke na roadmapi; 2) robot se pomera od ciljne konfiguracije do tačke na roadmapi; 3) dve tačke se spajaju koristeći linije na roadmapi. Roadmapa mora da predstavlja sve topološki različite dopustive putanje u konfiguracionom prostoru jer inače algoritam za planiranje kretanja nije kompletan. U najpoznatije roadmap metode spadaju: Voronoi dijagram, graf vidljivosti, freeway metod, metoda siluete (silhouette) i subgoal mreže (subgoal networks) (Slika 4.2).

Graf vidljivosti daje tačno rešenje za slučaj tipa problema tačkasti robot, a kompleksnost u tom slučaju mu je kvadratna. Međutim, mana mu je što je primenjiv samo za dvodimenzionalna okruženja, jer u složenijem konfiguracionom prostoru rešenje pripada klasi NP-teških. Ovaj pristup se zasniva na činjenici da najkraća putanja dodiruje mnogougaone prepreke na njihovim čvorovima, i pravi mrežu (roadmap) linija koje povezuju svaki čvor sa ostalim čvorovima koji su vidljivi iz njegove pozicije. Primenom grafa vidljivosti optimalna putanja je vrlo često blizu prepreka, što u slučaju problema koji imaju neodređenost u smislu tačnog položaja prepreka nosi određeni rizik.

Imajući u vidu da je u slučaju putanje minimalne udaljenosti od prepreka veoma teško kontrolisati i obezbediti da ne dođe do kolizija, predloženi su mnogi roadmap pristupi bazirani na skeletonu, među kojima je najpoznatiji Voronoi pristup. Voronoi dijagram je vrsta grafa koji se koristi kao generalno rešenje problema bliskosti u 2D okruženju. On se može opisati na sledeći način: ako je dat skup S od n tačaka u ravni, potrebno je za svaku tačku s iz skupa S odrediti region koji se sastoji od svih tačaka u ravni koje su bliže tački s od bilo koje druge tačke s' iz S. Voronoi pristup pravi skeleton koji je maksimalno udaljen od prepreka, i nalazi putanju minimalne udaljenosti koja prati taj skeleton. Algoritam radi u dvodimenzionalnom prostoru i njegova kompleksnost je $O(NlogN)$. Hijerarhijski Voronoi graf je pokušaj da se ovaj

pristup generalizuje i primenjuje za višestruke dimenzije. Freeway metod takođe pravi



(a) Graf vidljivosti

(b) Voronoi dijagram

(c) Subgoal mreže

(d) Metoda siluete

Slika 4.2 Roadmap metode [115]

skeleton koji je daleko od prepreka tako što slobodni prostor popunjava cilindrima.

Dekompozicija ćelija je drugi pristup i može da bude precizna (exact cell decomposition) i približna (approximate cell decomposition). Kod precizne ćelijske dekompozicije se slobodni konfiguracioni prostor razlaže u skup konveksnih mnogouglova, koji se međusobno povezuju putem grafa. Metodama pretraživanja grafa se zatim nalazi

optimalna putanja, a među najčešće korišćenim metodama pretraživanja je Dijkstra algoritam. U najpoznatije metode koje pripadaju preciznoj dekompoziciji spadaju: trapezoidna dekompozicija, dekompozicija bazirana na kritičnoj krivoj, cilindrična algebarska dekompozicija i metod povezanih lopti u slobodnom prostoru. Trapezoidna (vertikalna dekompozicija) deli slobodan prostor na trapezoidne regione koji ne dele prepreke. Nakon toga se formira roadmap spajanjem srednjih tačaka (midpoints) susednih trapezoida, i na tako dobijeni graf se primenjuje algoritam pretraživanja. Ovaj pristup je pogodan za primenu kod problema tačkastih vozila i kompleksnost mu je $O(NlogN)$. Za kruta vozila koja imaju mogućnost rotacije primenjuje se pristup baziran na kritičnoj krivoj. Ova dekompozicija se obavlja tako što se slobodan prostor deli na kritične i nekritične regione, tako da su granice ovih regiona deo po deo polinomijalne krive. Regioni se povezuju u graf koji se zatim pretražuje poznatim metodama. Kompleknost pristupa je $O(N^2logN)$. U metode precizne ćelijske dekompozicije spadaju još i: cilindrična algebarska dekompozicija, koja predstavlja proširenje dekompozicije bazirane na kritičnoj krivoj za slučaj trodimenzionalnih problema, i metod povezanih lopti u slobodnom prostoru (connected balls in free space). U metode približne dekompozicije ćelija spadaju pravougaona dekompozicija i $2m$ stablo dekompozicija. Pravougaona dekompozicija deli slobodan prostor u pravougaone regione tako da svaki od njih može da bude kompletno ispunjen (crn), delimično ispunjen (siv) ili kompletno prazan (beo). Kvadratni ili kockasti grid su najčešće korišćeni pristupi, a za pretraživanje se obično koriste A* ili D* metode. $2m$ stablo dekompozija se u poslednje vreme sve češće koristi zato što smanjuje broj potrebnih tačaka za predstavljanje prepreka u odnosu na ostale metode dekompozicije.

Treći pristup se naziva pristup potencijalnog polja, gde se na osnovu informacija o preprekama i ograničenjima određuje potencijalna funkcija, a zatim se primenom optimizacione metode najstrmijeg spusta (steepest gradient descent) na toj funkciji vrši generisanje putanje. Koncept potencijalnog polja je uveden 1986. god. i njime je definisana potencijalna funkcija kao diferencijabilna realna funkcija čija se vrednost može shvatiti kao energija, tako da je njen gradijent sila. Gradijent navedene potencijalne fukcije je vektor koji pokazuje pravac koji maksimizira potencijalnu energiju. U pristupu potencijalnog polja mobilni robot se posmatra kao tačka u prostoru koja se nalazi pod uticajem veštačkog potencijalnog polja U, pri čemu lokalne varijacije

polja daju informacije o strukturi slobodnog prostora. Potencijalna funkcija se može definisati kao zbir privlačnog potencijala, koji vuče robota ka krajnjoj konfiguraciji i odbojnog potencijala, koji robota odbija od prepreka. Pristup potencijalnog polja je zbog male računarske kompleksnosti metod koji se i dalje često koristi za rešavanje problema planiranja kretanja robota, pogotovu u slučaju kada je stepen slobode veliki [106]. U metode koje pripadaju pristupu potencijalnog polja spadaju: potencijalno polje sa gradijentnim metodom (gradient descent), pretraživanje vođeno potencijalnim poljem, harmonijske potencijalne funkcije, kontinualni Dijkstra, ekspanzija talasnog fronta i ekspanzija talasnog fronta sa skeletonom. Potencijalno polje sa gradijentnim metodom (virtuelno polje sile) je originalna i najstarija metoda iz ove grupe. Ciljnoj poziciji se dodeljuje decay funkcija sa minimalnom negativnom vrednošću, a svakoj prepreci se dodeljuje posebna decay funkcija sa pozitivnom maksimalnom pozitivnom vrednošću. Vrednosti ovih funkcija se zatim sabiraju i tako se dobija ukupno potencijalno polje. S obzirom da se prethodni metod može zaustaviti u lokalnom minimumu, umesto njega se često koristi pretraživanje vođeno potencijalnim poljem. Kontinualna Dijkstra metoda se koristi za dvodimenzionalne probleme i radi tako što deli prostor u mnogouglove vidljivosti (visibility polygons), dok je ekspanzija talasnog fronta verzija prethodne metode koja se može koristiti i za višedimenzionalne probleme, i veoma je slična kompletnom pretraživanju grida korišćenjem dinamičkog programiranja.

U četvrtu grupu bi se mogle svrstati sve ostale heurističke ili hibridne metode, kao što su fazi logika, neuralne mreže, slučajna stabla, probabilističke roadmape i prirodom inspirisane metaheuristike.

Uvidom u literaturu može se uočiti da je većina istraživanja posvećena analizi problema planiranja putanje u dvodimenzionalnom prostoru. Međutim, sredine u kojima se roboti kreću, kao što su šume, urbana ili podvodna sredina su obično nestrukturisane i pune nepredvidivih faktora, tako da su trodimenzionalni (3D) algoritmi neophodni. Današnji 3D algoritmi za planiranje putanje robota se mogu podeliti u sledeće kategorije: algoritmi bazirani na semplovanju (sampling based algorithms), optimalni algoritmi bazirani na čvoru (node based optimal algorithms), algoritmi bazirani na matematičkom modelu (mathematical model based algorithms), bioinspirisani algoritmi i multifuzioni algoritmi (multifusion based algorithms) [116].

Algoritmi bazirani na semplovanju zahtevaju informacije o celom okruženju, od-

nosno potrebno je napraviti matematički model za opis prostora. Obično se okruženje najpre deli (sempluje) na delove u obliku čvorova, ćelija ili na neki drugi način, a onda se radi pretraživanje u cilju dobijanja izvodljive putanje. Ovi algoritmi mogu da budu: aktivni, u koje spadaju rapidno istraživanje slučajnih stabala (rapidly exploring random trees, RRT), RRT dinamičkog domena (dynamic domain RRT, DDRRT), RRT-Zvezda, kao i veštačko potencijalno polje; i pasivni, od kojih su najvažniji 3D Voronoi, RRG, kao i probabilističke slučajne mape (probabilistic random maps, PRM) i modifikacije poput kPRM, sPRM i sl. Optimalni algoritmi bazirani na čvoru se mogu podeliti u tri grupe: u prvu spada Dijkstra algoritam, u drugu A*, Theta*, LPA*, a u treću D*, D*-Lite i sl. Algoritmi bazirani na matematičkom modelu su linearni algoritmi i optimalna kontrola. Ove metode prvo pomoću raznih jednakosti ili nejednakosti modeluju kinodinamička ograničenja koja služe kao ograničenja kriterijumske funkcije, a zatim se traži optimalno rešenje. Metod baziran na ravnosti (flatness), linearno programiranje mešavine celih brojeva (mixed integer linear programming, MILP), binarno linearno programiranje (binary linear programming, BLP) pripadaju ovoj grupi algoritama. Multifuzioni algoritmi u suštini predstavljaju kombinaciju više različitih metoda u cilju otklanjanja njihovih nedostataka i postizanja boljih performansi.

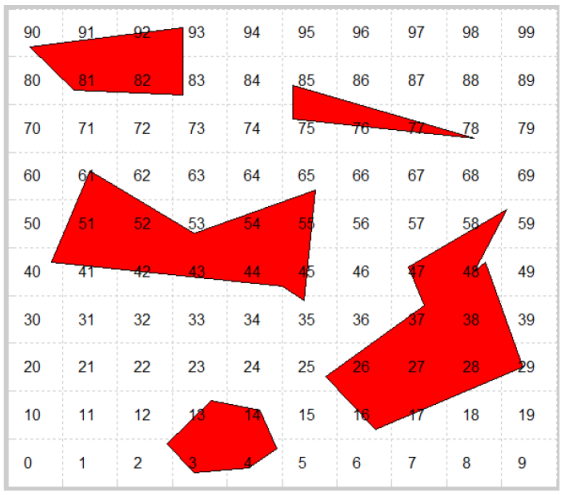
Većina tradicionalnih metoda se bazira na konstruisanju grafa kojim se predstavlja geometrijska struktura okruženja. Koristeći dati graf primenjuje se neka tehnika pretraživanja grafa kako bi se našla najbolja putanja koja sadrži početnu i krajnju tačku. Geometrijska struktura grafa zavisi od toga koji pristup se koristi za rešavanje problema. Najčešće korišćene metode za pretraživanje tako generisanog grafa su: A* pretraživanje, metod pretraživanja pomoću Voronoi dijagrama, matematičko programiranje, bi-level programiranje i D*-lite algoritam. Većina ovih algoritama koristi Eppstein-ov k-najbolji algoritam za nalaženje optimalne putanje. Njegova mana je to što ne uzima u obzir ograničenja kretanja UAV letelica, te se zbog toga ne može koristiti u realnim situacijama. Većina gore navedenih, tradicionalnih metoda ima generalni nedostatak koji se ogleda u zaustavljanju rešenja u lokalnom optimumu [117].

4.4 Planiranje putanje kao optimizacioni problem

Planiranje putanje se može razumeti kao višeciljni optimizacioni problem koji uključuje brojne zahteve i ograničenja. Nažalost, ovi ciljevi i zahtevi su međusobno neretko protivrečni, te se prilikom rešavanja problema mora praviti određeni kompromis. Problem planiranja putanje je težak optimizacioni problem tako da ne postoje determinističke metode za njegovo rešavanje u razumnom vremenu. Prilikom rešavanja problema planiranja putanje primenom prirodom inspirisanih optimizacionih algoritama neophodno je preduzeti sledeće korake: usvojiti pogodan matematički model okruženja, napraviti odgovarajuću ciljnu funkciju, i izabrati optimizacioni algoritam. Postoji više matematičkih modela za predstavljanje okruženja, od kojih su najpoznatiji: vektorski model, gde se prepreke predstavljaju pomoću mnogouglova; model mreže ćelija (grid, occupancy cell); i model grafa, od kojih su najpoznatiji MAKLINK graf, Voronoi graf (Voronoi dijagram) i graf vidljivosti.

MAKLINK graf služi za modelovanje slobodnog prostora u kojem se kreće mobilni robot. Ovaj pristup radi pod sledećim pretpostavkama: 1) robot se kreće u ograničenom 2D okruženju; 2) okruženje je mnogougaonog oblika i sadrži prepreke istog oblika; 3) granice prepreka su proširene za vrednost jednaku polovini dužine ili širine robota na koju je dodato minimalno rastojanje među relevantnim senzorima. Na ovaj način se robot u nastavku rada algoritma može posmatrati kao tačka. U modelu MAKLINK grafa svaka stranica mnogougla koji predstavlja prepreku je obuhvaćena sa nekoliko slobodnih MAKLINK linija koje se mogu definisati na sledeći način [118]: 1) krajnje tačke linije su dva čvora različitih prepreka, ili je jedan čvor na prepreci, a drugi na granici okruženja; 2) slobodna MAKLINK linija ne može da preseca nijednu prepreku. Grid model se može realizovati na dva načina: kao X-Y koordinatna ravan ili kao pravilno numerisani grid (orderly numbered grid), pri čemu je ovaj drugi pristup dominantan u literaturi. U pravilno numerisanom gridu prazne ćelije označavaju prostor gde robot može slobodno da se kreće, a sive ćelije prostor koji sadrži prepreke, pri čemu je veličina sivog dela određena veličinom prepreke na koju je dodata vrednost sigurnog odstojanja u skladu sa veličinom robota. Na ovaj način se potencijalna putanja sastoji od segmenata koji povezuju ćelije grida sa različitim brojevima i može se izraziti kao niz brojeva ćelija grida koje robot zauzima tokom kretanja duž putanje.

Nakon konstrukcije pogodnog matematičkog modela okruženja u kojem se robot



Slika 4.3 Grid model okruženja sa obeleženim mnogoogaonim preprekama [119]

kreće, potrebno je napraviti matematički model putanje u skladu sa izabranom metodom za rešavanje problema njenog planiranja. U samom jezgru problema planiranja putanje je uspostavljanje i definisanje efikasnog modela za predstavljanje ciljne funkcije tj. funkcije troškova putanje zato što je vrednost ove funkcije kriterijum za evaluaciju putanje; što je vrednost funkcije manja, putanja je bolja, i obrnuto. Ciljna funkcija mora biti tako definisana da uvaži sve ciljeve, zahteve i ograničenja.

Kada se rešava višeciljni optimizacioni problem, tada postoji više rešenja koja optimizuju jedan cilj na račun drugog. Ovakva rešenja se nazivaju Pareto optimalni skup ili front, i svako rešenje iz ovog skupa se može prihvatiti kao validno. Takođe, postoje takvi problemi gde se prednost jednom rešenju u odnosu na drugo daje na osnovu nenumeričke kvalitativne informacije ili heuristike. U slučaju problema planiranja putanje to može biti: manje promena smera kretanja, kretanje kroz manji broj čvorova grida, više skretanja u desno nego u levo i sl. Imajući u vidu brojne zahteve i ograničenja, veoma je teško napraviti matematički model za problem planiranja putanje koji bi bio rešavan metodama optimizacije. Ako je optimizacioni problem takav da se ne mogu zadovoljiti sva ograničenja, onda se on naziva problem zadovoljavanja ograničenja (constraint satisfaction problem, CSP). Može se desiti da je broj ograničenja u problemu planiranja putanja preveliki da bi se on algoritamski uspešno rešio u razumnom vremenu. Tada se prihvataju ili preferiraju rešenja koja nisu u obliku Pareto optimalnog skupa, već imaju CSP formu.

5 PLANIRANJE PUTANJE ROBOTA PRIMENOM PRIRODOM INSPIRISANIH METAHEURISTIKA

Poslednjih godina je predložen veliki broj prirodom inspirisanih algoritama za problem planiranja putanje mobilnih robota. U nastavku je dat pregled istraživanja koja su imala za cilj rešavanje problema planiranja putanje robota, UAV i UCAV letelica primenom ovih algoritama, sa posebnim naglaskom na algoritme inteligencije rojeva.

U [120] je primenjen algoritam simuliranog kaljenja na problem planiranja putanje mobilnog robota u dvodimenzionalnom prostoru, pri čemu su uzete u obzir tri reprezentacije putanje: linearna, Bezier-ova kriva i splajn interpolirana kriva. Cilj je naći optimalnu putanju na kojoj nema kolizija, a kriterijumska funkcija je definisana na osnovu dužine putanje. Algoritam simuliranog kaljenja je korišćen u kombinaciji sa tradicionalnom metodom veštačkog potencijalnog polja da bi se ublažio njegov nedostatak koji se odnosi na zaustavljanje u lokalnom minimumu [121], [122].

5.1 Evolucioni algoritmi

Sistem za planiranje putanje u realnom vremenu, koji se zasniva na evolucionom algoritmu je predložen u [123]. Planer ima mogućnost rada u okruženju gde su promene nepredvidive, i uzima u obzir različita ograničenja problema kao što su minimalna dužina putanje i altutuda leta, maksimalni ugao okretanja i fiksni vektor prilaska tj. određeni ugao prilaska ciljnoj poziciji, a takođe može biti korišćen za rad u sistemu sa jednim i sa više vozila. U [124] se predlaže algoritam za planiranje putanje više UAV vozila u realnim uslovima. Predloženo rešenje se zasniva na evolucionim algoritmima primenom pristupa višestruke koordinisane koevolucije agenata. Dobijene putanje su izračunate na osnovu osobina realnih UAV, karakteristika terena, radara i projektila, i strukturirane su po različitim nivoima prioriteta u zavisnosti od misije letelice. Planer radi u offline i online modu kako bi se uspešno nosio sa nepredvidivim rizicima u toku leta i po potrebi radio ponovno izračunavanje delova putanje. Evolucioni pristup se koristi i u [125] za nalaženje putanje u kompleksnom okruženju, kao i u [126].

5.1.1 Genetski algoritmi

Pehlivanoglu [127] predlaže multifrekventni vibracioni genetski algoritam (mVGA) za rešavanje problema planiranja putanje UAV letelica. Autor je implementirao novu strategiju mutacije, a u inicijalnim fazama rada algoritma su primenjene metode klasterovanja i Voronoi dijagram. Operator mutacije se primenjuje dva puta nakon operatora ukrštanja. Prva primena se obavlja radi obezbeđivanja globalnog slučajnog diverziteta u populaciji, a druga ima za cilj lokalni diverzitet u susedstvu elitne individue. Inicijalna populacija je predstavljena kao zbir slučajne populacije i ostatka populacije. Slučajna populacija se sastoji od individua koje se generišu na slučajan način, u skladu sa ograničenjima modela terena koji se primenjuje. Ostatak populacije je generisan primenom Voronoi dijagrama. Modelovanje putanje je urađeno pomoću Bezier-ovih krivih zato što one omogućuju izračunavanje glatkih dinamičkih putanja. Ciljna funkcija je napravljena tako da ostvari tri zahteva: minimizaciju dužine putanje, održavanje glatke trajektorije i održavanje bezbednog rastojanje letelice od terena, i realizovana je kao linerana kombinacija ova tri zahteva. Predloženi algoritam je upoređivan sa tri druga konkurentna GA bazirana algoritma, u dva različita 3D okruženja, i to: sinusoidalni i gradski model. Rezultati su utvrdili superiornost predloženog rešenja u smislu potrebnog vremena izračunavanja.

U [128] se opisuje poboljšana verzija genetskog algoritma za planiranje putanje mobilnih robota. Modifikacije se odnose na operacije ukrštanja, mutacije i brisanja osnovnog GA algoritma, kao i na integraciju kontrolnog algoritma na bazi fazi logike kojim se samoadaptivno podešavaju verovatnoće operacija ukrštanja i mutacije. Za matematičko modelovanje problema se koristi pravilno numerisan grid. Jedna od mana GA algoritma u njegovoj primeni na određivanje optimalne putanje mobilnih robota je operator mutacije, što ima za rezultat generisanje neizvodljivih putanja. U tom smislu su autori u [129] napravili novi GA algoritam sa modifikovanim operatorom mutacije za planiranje putanje mobilnih robota u dinamičkom okruženju sa preprekama. Model okruženja je pravilno numerisan grid. Cilj algoritma je minimizacija putanje, a kriterijumska funkcija je napravljena tako da se koristi metod kazne (penalty method) za neizvodljive putanje, tj. putanje koje sadrže prepreke. Kazna se dodaje na ukupnu dužinu putanje, i njena vrednost je veća od dužine najveće generisane putanje. Računarski efikasan algoritam za planiranje putanje mobilnog robota

gde su kriterijumi dužina putanje i bezbednost je dat u [130]. Okruženje je predstavljeno gridom, a za kreiranje numeričkih potencijalnih polja, kako za ciljne tačke tako i za prepreke, se koristi metod talasnog fronta. AL-Taharwa i dr. [131] su predložili primenu GA algoritma za planiranje putanje mobilnog robota u statičkom okruženju koje je modelovano gridom, pri čemu se analiziraju dva slučaja kretanja: bez prepreka i sa preprekama.

Primena genetskog algoritma sa elitističkim nedominirajućim sortiranjem na rešavanje višeciljnog problema planiranja putanje vozila je prikazana u [132]. Predložene su četiri različite šeme za predstavljanje putanje, a okruženje je predstavljeno u obliku grida. Ciljevi su definisani tako da se optimizuje dužina putanje i bezbednost, a kao sekundarni cilj pojavljuje se i glatkost putanje. Dodatno, obavljena je i analiza nekoliko važnih pitanja koja se tiču problema optimizacije putanje kao što su rešavanje ograničenja, identifikacija efikasne šeme za reprezentaciju putanje, razlike između jednociljne i višeciljne optimizacije putanje, te evaluacija predloženog algoritma na veoma velikim gridovima gde su prepreke gusto raspoređene. Cheng i dr. [133] su predložili imuni genetski algoritam (immune genetic algorithm, IGA) sa elitističkim pristupom za planiranje putanje UAV letelica. Modifikovani algoritam koristi imuni operator i koncentracioni mehanizam, čime se smanjuju inherentni nedostaci originalnog GA algoritma koji se tiču preuranjene i spore konvergencije. Nikolos i dr. [134] su opisali algoritam koji predstavlja kombinaciju više genetskih algoritama za offline/online planiranje putanje UAV letelica u 3D, pri čemu se za modelovanje putanje koriste B-Spline krive čije su koordinate kontrolnih tačaka veštački geni evolucionog algoritma.

Trodimenzionalni planer putanje UAV letelica realizovan korišćenjem višeciljnog genetskog algoritma nedominirajućeg sortiranja (Non-dominated sorting genetic algorithm II, NSGA II) je dat u [135]. Okruženje je matematički predstavljeno u obliku meš 3D površi (meshed 3D surface). Algoritam ima dva cilja: minimizaciju dužine putanje i maksimalnu marginu bezbednosti, a putanja koja se generiše je predstavljena B-Spline krivom, tako da su kontrolne tačke B-Spline krive promenljive odlučivanja genetskog algoritma. Rešenja za problem planiranje putanje robota i autonomnih vozila koja se zasnivaju na GA algoritmu i njegovim modifikacijama data su i u [136], [137], [138], [139] i [140].

5.1.2 Diferencijalna evolucija

Zhang i Duan [141] su opisali poboljšanu verziju DE algoritma za rešavanje globalnog problema planiranja putanje UAV letelica u 3D okruženju. Autori su dizajnirali svoje rešenje imajući u vidu dva cilja: kratku putanju i nisku altitudu leta. Dodatno, uzeta su u obzir i mnoga ograničenja koja postoje u realnim uslovima leta, kao što su: maksimalni ugao okretanja, maksimalni nagib uzleta/poniranja, konfiguracija terena, zabranjena zona leta, zona pretnji i opseg mape leta. Matematičko predstavljanje okruženja i putanje je isto kao u našem istraživanju u 6.1, sa razlikom što postoji i treća koordinata visine. Predloženi algoritam koristi dinamičku strategiju za obezbeđenje glatkosti trajektorije, a prostor pretraživanja je ograničen kako bi se proces učinio efikasnijim. Ciljna funkcija se predstavlja zbirom dve komponente, J_L i J_H, koje se odnose na dužinu putanje i altitudu leta, respektivno. Dužina se računa kao zbir segmenata, gde se svaki segment računa kao euklidsko rastojanje između dve tačke. Poželjno je da UAV letelica leti na nižim altitudama jer tako zbog maskiranja terena bolje izbegava radare. Troškovi altitude leta se mogu izračunati po formuli $J_H = \int_{P_{UAV}} H_p dl$, gde je $H_p = 0$, ako je z_k manje od 0, a u suprotnom ima vrednost z_p (vrednost altitude u tački p). Optimizacioni problem se posmatra kao ograničeni, gde su ograničenja koja se tiču maksimalnog ugla okretanja, nagiba uzleta/poniranja, i ograničenja terena predstavljena nejednakostima (inequality constraints), a ograničenja koja se odnose na zabranu letenja u zabranjenim oblastima, zonama pretnji i let van opsega mape predstavljena jednakostima (equality constraints). Radi efikasnijeg rešavanja problema planiranja putanje autori obavljaju transformaciju optimizacionog problema sa ograničenjima u problem bez ograničenja primenjujući tehniku α nivo poređenje (α level comparison). Ovom tehnikom se za ograničenja uvodi α nivo zadovoljavanja (α satisfaction level) kojim se definiše koliko dobro potencijalna rešenja zadovoljavaju ograničenja, npr. ako je nivo zadovoljavanja manji od 1, rešenje nije izvodljivo. Dakle, selekcioni operator originalnog DE algoritma je modifikovan primenom tehnike α nivo poređenja i tako je kreiran algoritam diferencijalne evolucije sa nivoom poređenja (DELC). Dodatno, autori predlažu modifikovanu strategiju za ažuriranje vrednosti α koristeći sigmoidnu funkciju. Predloženi DELC algoritam je putem numeričkih eksperimenata upoređivan sa šest postojećih optimizacionih algoritama sa ograničenjima i pet metoda baziranih na penalty funkcijama

(koje tradicionalno služe za rad sa ograničenjima). Rezultati su pokazali da DELC algoritam pokazuje dobre performanse u smislu kvaliteta, robustnosti i mogućnosti zadovoljavanja ograničenja. Implementacija 3D planera trajektorija za UAV letelice koji koristi poboljšanu verziju DE algoritma opisana je u [142]. U cilju otklanjanja nedostataka originalnog DE algoritma koji se odnosi na problem preuranjene i spore konvergencije, autori su predložili primenu haotičnog pretraživanja u operatoru mutacije. Kriterijumska funkcija (funkcija troškova) se računa kao zbir troškova koji se odnose na dužinu i visinu putanje, i pretnje duž putanje.

Cilj rada [143] jeste ispitivanje mogućnosti korišćenja DE algoritma za dizajn offline planera u statičkom morskom okruženju koji ima mogućnost generisanja 2D putanja za koordinisanu navigaciju više UAV letelica. Polazeći od pretpostavke da svaka UAV kreće iz različite inicijalne lokacije, algoritam treba da generiše putanje sa određenom poželjnom distribucijom brzina duž istih, tako da letelice stignu na destinaciju uz izbegavanje međusobnih kolizija i sudara sa preprekama, kao i uz zadovoljavanje ciljeva i ograničenja putanje i koordinacije. Problem kooperativnog problema planiranja putanje više robota je analiziran i u [144]. Predloženo rešenje koristi paralelne algoritme diferencijalne evolucije, a autori su uporedili dva moguća pristupa rešavanju problema: centralizovani i distribuirani. Rezultati su pokazali da je distribuirani pristup bolji od centralizovanog, kao i da je konkurentan PSO implementaciji planera.

5.2 Optimizacija rojevima čestica

U [145] je data preliminarna studija multi-swarm scenarija deljenja za optimizaciju rojevima čestica i njegova primena na rešavanje problema planiranja putanje UAV letelica. Radi realizacije predloženog algoritma koji radi sa više rojeva, autori su uključili dva nova procesa: ukrštanje rojeva (swarm crossover) i upravljanje rojevima (swarm manager). Ukrštanje rojeva se implementira sledeći koncept ukrštanja iz genetskih algoritama. Prvo se selektuju dva roditelja iz skupa od R globalno najboljih pozicija (R je broj rojeva), a zatim se na slučajan način od izabranih roditelja biraju dve tačke ukrštanja; njihove informacije koje su iza izabranih tačaka se menjaju, i proizvode se nova dva potomka koji proširuju ili sužavaju dimenziju pretraživanja. Potomci se zatim evaluiraju i porede se njihove fitnes vrednosti kako bi se izabrao bolji koji će poslužiti za generisanje novog roja. Radi uklanjanja neželjenih čestica i rojeva, kao i

radi sprečavanja nerazumnog rasta populacije uvodi se menadžer rojeva koji u svakoj generaciji briše čestice sa najgorom fitnes vrednošću. Ovaj pristup pozitivno utiče na vreme izračunavanja algoritma. Predloženi metod se pokazao dobrim u nalaženju pogodne i izvodljive putanje u 3D modelu leta.

U [109] se za rešavanje problema planiranja putanje mobilnih robota u dvodimenzionalnom okruženju sa dinamičkim preprekama predlaže modifikovani membranom inspirisani algoritam baziran na PSO (membrane inspired particle swarm optimization, mPSO). Problem planiranja putanje se posmatra kao višeciljni optimizacioni problem gde se razmatraju tri cilja: rastojanje, sigurnost i glatkost. Kako bi se realizovao kompromis između međusobno suprostavljenih ciljeva i poboljšala konvergencija algoritma uvode se point repair algoritam i glatki pristup. Stepen sigurnosti je suma devijacionih stepena C_i $(i = 1, 2, \ldots, N)$ između bilo kojeg segmenta putanje i najbliže prepreke i definiše se na sledeći način:

$$SD = \sum_{i=1}^{n-1} C_i = \begin{cases} 0, d \geq \alpha \\ \sum_{i=1}^{n-1} e^{\lambda - d_i}, d < \alpha \end{cases} \tag{5.1}$$

mPSO algoritam koristi dinamičku strukturu membrane, gde se menjaju OLMS i D-OLMS, kako bi se uredila populacija čestica koje predstavljaju potencijalne putanje robota, i specificirala razna pravila, kao što su podela membrane, pravila transformacije i komunikacije, i razlaganje membrane. Dimenzija PSO čestica se dinamički menja u toku rada algoritma. Point repair algoritam se koristi za pretvaranje neizvodljivih putanja u izvodljive. Algoritam glatkosti uklanja nepotrebne čvorove i na taj način smanjuje dimenziju čestica. Takođe, koristi se i tehnika podešavanja smera kretanja kako bi se ubrzala konvergencija algoritma. Izvršeno je poređenje predloženog algoritma sa PSO i GA u različitim okruženjima koja podrazumevaju tri grid modela i pet vrsta prepreka, čime je ustanovljena efikasnost mPSO algoritma.

Cilj istraživanja u radu [146] je bio da se analizira problem planiranja putanje inteligentnih vozila u dinamičkom okruženju. Predloženi metod se bazira na PSO algoritmu i metodu dinamike ponašanja (behaviour dynamics method). Metod dinamike ponašanja je zadužen za generisanje problema kompetitivnog ponašanja, dok se za poboljšanje koordinacije ponašanja koristi PSO pristup. Dinamički model se pravi

na osnovu promenljivih ponašanja i šablona ponašanja koji inteligentno vozilo sledi. U promenljive ponašanja spadaju ugao kretanja (heading angle) i brzina inteligentnog vozila. Cilj opšteg ponašanja vozila se sastoji iz dva elementa: ponašanje dostizanja ciljne pozicije i ponašanje izbegavanja prepreka. Za svaku od ova dva elementa se konstruišu odgovarajući modeli, i to posebno za svaku od varijabli ponašanja. Nakon definisanja različitih dinamičkih modela, predloženi metod zahteva da se uradi koordinacija ovih modela ponašanja fuzijom nekoliko varijanti ponašanja, tj. da se napravi fuzija varijabli preko odgovarajućih težinskih koeficijenata i da se nakon toga realizuje planiranje putanje vozila. PSO optimizacioni algoritam se ovde primenjuje kako bi se uradila optimizacija pomenutih težinskih koeficijenata, pri čemu se fitnes funkcija definiše tako da odslikava vezu koja postoji između vozila i okruženja, tj. prepreka. Odgovarajuća vrsta ponašanja treba da ima prednost u zavisnosti od dinamike kretanja: ako je prepreka blizu vozila, ponašanje izbegavanje prepreka treba da ima prednost; i obrnuto, ako je vozilo daleko od prepreke, ponašanje dostizanja ciljne pozicije dominira. Rezultati simulacija su pokazali da je predloženi metod efikasan u pogledu pouzdanosti i performansi u realnom vremenu, i da je PSO algoritam dobro rešenje za probleme koordinacije ponašanja.

Trodimenzionalni algoritam za planiranje putanje UAV letelica koji se zasniva na operatoru odlučivanja adaptivne osetljivosti (adaptive sensititivity decision operator) u kombinaciji sa PSO algoritmom je dat u [147]. Matematičko modelovanje u predloženom metodu koristi kardinalne splajn funkcije dok se za definisanje determinističkih pretnji kao što su radari, artiljerija i projektili koristi cilindrični model. Matematička reprezentacija putanje je urađena kao na način koji se veoma često koristi u literaturi, a koji smo mi primenili u našem istraživanju u 6.1. Obeleže se početna i krajnja tačka, koje se spajaju tako da se formira duž. Zatim se data duž vertikalnim linijama normalnim na duž deli u $M + 1$ segmenata, a tačke preseka se nazivaju waypoints. Oblasti pretnji su predstavljene krugovima, tako da je osetljivost na pretnju data verovatnoćom koja je obrnuto proporcionalna udaljenošću od centra pretnje; što je veća udaljenost segmenta putanje leta od centra pretnje manja je verovatnoća da će pretnja predstavljati opasnost za letelicu. Ciljevi planiranja putanje su izbegavanje svih oblasti pretnje uz najmanje troškove, najkraći ukupan put i najnižu moguću altitudu. Matematička reprezentacija problema zahteva transformaciju koordinata kako

bi se ubrzao algoritam. Duž koja spaja početnu i krajnju tačku postaje nova apscisa, a nova ordinata se dobija rotiranjem stare u smeru suprotnom od smera kazaljke na satu za ugao θ između nove i stare apscise. Geometrijska veza između istih waypoint tačaka u novom i starom koordinatnom sistemu je data sledećom jednačinom:

$$\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_O - x_S \\ y_O - y_S \\ z_O - z_S \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ z_S \end{bmatrix} \tag{5.2}$$

gde indeks r označava novi, transformisani koordinatni sistem, O označava početnu tačku, a S krajnju tačku. Funkcija cilja se definiše na sledeći način:

$$J_{obj} = J_{hit} + J_{in} + J_{out} + J_{fall} + J_{dis} + J_{alti} + J_{ToD} + J_{lim} \tag{5.3}$$

gde je: J_{hit} – ukupan broj waypoint tačaka koje se nalaze unutar terena; J_{in} – stepen izloženosti pretnji UAV u datoj tački; J_{out} – broj tačaka izvan specificirane oblasti leta; J_{fall} – ukupni efekat pretnji; J_{dis} – dužina putanje; J_{alti} – akumulirana razlika između UAV altitude i terena u datoj tački; J_{ToD} – prosečna udaljenost waypoint tačaka do destinacije; J_{lim} – komponenta koja se odnosi na ograničenja u prostoru pretraživanja. Predloženi algoritam se razlikuje u odnosu na standardni PSO po tome što se formira oblast odlučivanja adaptivne osetljivosti sa ciljem poboljšanja performansi algoritma. Ova oblast omogućuje da se sa visokom verovatnoćom odrede potencijalne lokacije čestica, a da se ostala kandidatska rešenja izbrišu. Da bi se izbegla preuranjena konvergencija, prostor pretraživanja se drži u određenim granicama, a relativna usmerenost čestica od tekuće lokacije omogućuje poboljšanje pretraživanja. Primenom indeksa prave putanje (straight line rate, SLR), koji se definiše kao odnos između dužine kandidatske putanje i dužine prave linije između startne i ciljne pozicije, i uparenog T-Testa, autori su pokazali da je predloženi metod bolji od standardnog PSO, algoritma svica i genetskog algoritma.

Zhang i dr. [148] predlažu višeciljni algoritam za planiranje putanje robota u neizvesnom okruženju, koji se zasniva na PSO algoritmu optimizacije. Neizvesno okruženje podrazumeva postojanje pretnji čiji se položaj ne može odrediti sa preciznošću. Matematička formulacija putanje je ista kao u našem istraživanju u 6.1, a za

kriterijume performansi su uzeti dužina putanje i stepen pretnje uz uslov da putanja mora da bude bez kolizija. Dužina putanje se računa na standardan način, kao suma euklidskih rastojanja između tačaka na putanji, odnosno, kao zbir segmenata putanje. Kad je reč o stepenu pretnje, autori prvo analiziraju slučaj gde pretnje, tj. izvori opasnosti, imaju statičke pozicije, kada se uticaj pretnje predstavlja linearnom fazi funkcijom, a zatim se posmatra slučaj kada su pozicije izvora opasnosti neodređene. Na ovaj način problem planiranja putanje sa neodređenim izvorima pretnji se svodi na ograničeni biobjektivni (dvociljni) optimizacioni problem sa neodređenim koeficijentima. Autori predlažu modifikaciju originalnog PSO algoritma u delu koji se odnosi na ažuriranje čestica, koja se zasniva na slučajnom semplovanju i uniformnoj mutaciji. Poznato je da se kod višeciljnih optimizacionih problema sa ograničenjima evaluacija rešenja radi ne samo na osnovu vrednosti fitnes funkcije nego i na osnovu stepena kršenja ograničenja. U ovom radu se za evaluaciju čestica predlaže poboljšana verzija veze ograničenog dominiranja (constrained dominance relationship) koja se bazira na nepreciznoj vezi dominiranja (imprecise dominance relationship). Uvode se stepeni narušavanja ograničenja (constrained-violated degrees) koji se računaju na osnovu broja kolizija čestica sa preprekama tj. izvorima opasnosti. Dodatna modifikacija se odnosi na uvođenje arhive neizvodljivih rešenja (infeasible archive) koja se koristi za čuvanje neizvodljivih nedominiranih rešenja koja mogu poslužiti kao most ka istraživanju izolovanih izvodljivih rešenja.

PSO sa kodiranjem faznog ugla i kvantnim ponašanjem (phase angle-encoded quantum behaved PSO, θ-QPSO) je dizajniran i primenjen na rešavanje problema planiranja putanje UAV letelica u trodimenzionalnom okruženju u [149]. Umesto vektora položaja i brzine, u θ-PSO se koriste vektori faznog ugla i njegovog inkrementa, a u θ-QPSO samo vektor faznog ugla, čime se smanjuju troškovi izračunavanja i trošenja memorijskih resursa. Funkcija troškova putanje uzima u obzir dužinu putanje, pretnje, ugao okretanja, ugao uspona/poniranja, i visinu tj. altitudu.

U [150] je predstavljen algoritam za planiranje putanje izviđanja UAV koji se zasniva na PSO algoritmu. Funkcija cilja se računa kao odnos troškova izviđanja i efekata izviđanja, pri čemu se troškovi računaju kao zbir troškova pretnje i potrošnje goriva. Efekti izviđanja su povezani sa vrednošću mete tako da efektivna putanja izviđanja predstavlja količinu informacija koje o meti prikupi UAV.

Autori su u [151] koristili MAKLINK graf za matematičko modelovanje problema, zatim su primenili Dijkstra algoritam za nalaženje najkraćeg rastojanja u tako formiranom grafu, a za optimizaciju putanje su primenili modifikovani PSO algoritam. Planer putanje baziran na poboljšanom PSO u koji je integrisan heuristički mehanizam za pretnje je predložen u [152]. Heurističke informacije se koriste u formuli za ažuriranje brzine čestica i na taj način usmeravaju kretanje čestica i poboljšavaju performanse pretraživanja PSO algoritma. Za smanjene dimezija čestica se koristi površ minimalnog rizika, a algoritam primenjuje online pristup u planiranju putanje kako bi se suočio sa nepredvidivim pretnjama. Da bi se smanjila računarska kompleksnost postupka planiranja putanje autori u [153] predlažu poboljšani stohastički PSO algoritam koji ima visoku sposobonost eksploracije i primenju ga na planiranje putanje mobilnog robota u okruženju sa statičkim preprekama. Za razliku od ostalih metoda, predloženi algoritam obezbeđuje generisanje glatkih putanja.

Dva nova algoritma za planiranje putanje robota su predložena u [154]. Prvi algoritam ima kao cilj najkraću dužinu putanje i realizovan je kao hibrid PSO algoritma i probabilističkog roadmap metoda (PRM), tako da se PSO koristi kao globalni, a PRM kao lokalni planer. U drugom algoritmu se glatkost putanje pojavljuje kao cilj, a predloženo rešenje je kombinacija novog ili negativnog PSO (NPSO) i PRM metoda. NPSO radi tako što se smer pretraživanja određuje na osnovu položaja najgorih čestica, a ne na osnovu najboljih. PSO i PRM se kombinuju tako što se najbolje čestice dodaju kao pomoćni čvorovi slučajnim čvorovima koje generiše PRM. Rezultati su pokazali da je rešenje sa NPSO bolje. Gong i dr. su razvili algoritam za planiranje putanje robota u okruženju koje sadrži izvore opasnosti, a koji se bazira na višeciljnoj PSO optimizaciji [155]. Slobodan prostor je predstavljen mapom koja se sastoji od niza horizontalnih i vertikalnih linija, a ciljna funkcija uzima u obzir dužinu putanje i stepen opasnosti. Višeciljni PSO algoritam koji se koristi za generisanje optimalne putanje sadrži samoadaptivni mutacioni operator baziran na stepenu putanje blokirane preprekom. Dodatna modifikacija je postojanje arhive koja čuva neizvodljiva rešenja, pored već postojeće arhive izvodljivih rešenja, koja ima za cilj poboljšanje algoritma u delu koji se odnosi na eksploraciju.

U [156] je predstavljen algoritam za globalno planiranje putanje robota baziran na binarnom PSO algoritmu. Prepreke su predstavljene pomoću mnogouglova, a čvorovi

mnogougla su numerisani vrednostima od 1 do n. Dužina PSO čestice je n, a vrednost svake promenljive može biti 0 ili 1, u zavisnosti od toga da li je čvor na putanji ili ne. Algoritam koristi mutacioni operator kojim se sprečava preuranjena konvergencija. Ostali radovi gde se PSO algoritam i njegove modifikacije primenjuju u planiranju putanje su [157], [158], [159], [160].

5.3 Algoritam kolonije mrava

Tan i dr. su predložili metod za planiranje putanje koji koristi poboljšani Dijkstra algoritam i ACO optimizaciju [161]. Prvi korak je modelovanje okruženja korišćenjem MAKLINK grafa, zatim se primenjuje poboljšani Dijsktra algoritam za nalaženje suboptimalne putenje bez kolizija, a na kraju se vrši optimizacija ACO algoritmom. Na početku rada optimizacionog algoritma tačke putanje se nalaze na središtu MAKLINK linija, a cilj algoritma je da se optimizuje njihova pozicija na MAKLINK linijama. Moguća lokacija se može definisati na sledeći način: Neka su $P_0, P_1, P_2, \ldots, P_{d+1}$ tačke putanje dobijene Dijkstra algoritmom, gde je P_0 početna a P_{d+1} ciljna pozicija. Tada je moguća lokacija na MAKLINK liniji data sa: $P_i = P_{i1} + (P_{i2} - P_{i1}) * h_i, i = 1, 2, \ldots, d$. Kriterijumska funkcija L se može definisati sledećom jednačinom:

$$L = \sum_{i=1}^{d} length(P_i(h_i), P_{i+1}(h_{i+1})) \tag{5.4}$$

Cilj optimizacije je da se nađe niz parametara h_i tako da funkcija L ima minimalnu vrednost.

Primena ACO algoritma za planiranje putanje autonomnih podvodnih vozila je data u [162]. Za modelovanje se koristi graf vidljivosti baziran na grid modelu okruženja. Algoritam je poboljšan sa nekoliko novih pravila za ažuriranje feromona, a radi obezbeđenja glatkosti uvedena su dva operatora: operator odsecanja i operator umetanja (insertion-point operator). Optimizacija planiranja putanje podvodnih vozila primenom poboljšanog ACO algoritma (IACO) je data u [163]. Prostor je modelovan kvadrom odgovarajućih dimenzija, koji se zatim deli vertikalnim i horizontalnim ravnima u mrežu (grid) 3D oblasti. Algoritam koristi pristup odbijanja feromona (pheromone exclusion). Feromoni mrava u ovom algoritmu sadrže privlačni deo i odbojni deo. Mrava privlače sopstveni feromoni, a odbijaju ga feromoni drugih mrava. ACO algoritam je

predlagan za nalaženje optimalne putanje podvodnih vozila i u [164], [165].

Poboljšani ACO algoritam za planiranje putanje UAV koje lete na niskim visinama je predložen u [166]. Okruženje je predstavljeno grid mapom, a kriterijumska funkcija sadrži tri komponente: komponenta koja je povezana sa dužinom putanje tj. trošak devijacije od prave linije koja povezuje početnu i krajnju poziciju, kazna (penalty) ukoliko dođe do približavanja zonama opasnosti, i komponenta koja se odnosi na minimizaciju altitude leta. Algoritam ima dve modifikacije koje se odnose na adaptivno biranje sledećeg čvor prilikom kretanja mrava, i adaptivno podešavanje feromona. Parametar evaporacije feromona ima inicijalnu vrednost 0.1 i menja se adaptivno u toku rada algoritma. Pseudokod algoritma je prikazan u Algoritmu 5.1.

Algoritam 5.1: Poboljšani ACO algoritam za planiranje putanje UAV

1 Formiraj originalnu matricu feromona T;
2 M mrava se postavljaju u startnu poziciju;
3 Svaki mrav bira sledeći čvor u grid mapi na osnovu pravila diverzije, dostižući na kraju ciljni čvor i tako formira putanju;
4 Izračunati funkciju troškova svih generisanih putanja iz (3) i sačuvati optimalno rešenje;
5 Ažurirati feromone svakog mrava na osnovu funkcije troškova, a u skladu sa pravilom o ažuirarnju feromona;
6 Evaluirati optimalno rešenje i odlučiti da li je potrebno ažuriranje gena evaporacije feromona;
7 Ispitati da li je zadovoljen uslov prestanka rada algoritma, i ponavljanje koraka (1) – (6) ako nije;

Replaniranje koordinisane trajektorije više UAV letelica u dinamičkom i neodređenom okruženju primenom Max-Min adapativnog ACO algoritma je predloženo u [167]. U ovom algoritmu mravi iz svake podpopulacije uzimaju podatke iz svoje, ali iz drugih podpopulacija prilikom odlučivanja o sudarima. Koordinacija UAV podrazumeva: istovremeni dolazak na destinaciju, što zahteva određivanje procenjenog vremena dolaska grupe (estimated time of arrival, ETA), i izbegavanje sudara. Okruženje je podeljeno i predstavljeno u obliku dvodimenzionalne mreže (grid). Troškovi pretnje se računaju na isti način kao u našem istraživanju u 6.1. Algoritam uzima u obzir ograničenja kao što su: minimizacija ugla okretanja, ograničenja vremena koordinacije (sve UAV treba da stignu na odredište istovremeno) i izbegavanje sudara. Replaniranje putanje se sastoji iz tri dela: odlučivanje o koordinaciji (coordination decider),

planer trajektorije (trajectory planner) i obezbeđivanje glatkosti trajektorije (path smoother). Radi izbegavanja stagnacije pretraživanja uvodi se ograničavanje uticaja tragova feromona tako što se postavljaju eksplicitne granice (min i mix) za vrednosti tragova feromona, koje važe za sve feromone.

Chen i dr. [168] su predstavili modifikovani ACO algoritam za planiranje putanje UCAV u 3D okruženju. Najpre se daje rešenje za slučaj statičkog okruženja, a onda se predlaže planer putanje u dinamičkom 3D okruženju za UCAV letelice, kada je zbog nepredvidivih pretnji potrebna rekalkulacija putanje. Kriterijumska funkcija uzima u obzir troškove potrošnje goriva, troškove pretnji i troškove visine, a definisana su i sledeća ograničenja: 1) ograničenja uglova skretanja u horizontalnoj i vertikalnoj ravni; 2) najveća i najmanja visina leta (mala visina nosi rizik kolizije sa preprekama, a velika visina povećava verovatnoću detekcije od strane neprijateljskih radara); i 3) najbrži opseg leta (da letelica ne bi ostala bez goriva). Predložene su sledeće izmene originalnog ACO algoritma. Uvodi se težinska funkcija u izraz kojim se definiše verovatnoća tranzicije, kako bi se povećale brzina selekcije i verovatnoća selekcije kada je intenzitet feromona na putanji veliki. Takođe, kada je broj mrava na nekoj ruti veći od trećine ukupnog broja mrava, uvodi se aritmetički operator koji zamenjuje parametar evaporacije feromona, da bi se poboljšala verovatnoća lokalno najboljih rešenja. Kad je reč o replaniranju putanje, autori integrišu dva poznata pristupa: prvi, koji radi replaniranje svih segmenata putanje od tekućeg čvora od ciljnog čvora; i drugi, koji radi samo replaniranje segmenata putanje u oblasti pretnje. Mana prvog pristupa je dugo vreme izračunavanja, a drugog što je nova putanja samo lokalno najbolja. Autori predlažu da se prilikom detekcije nove pretnje vrši izbor odgovarajućeg prozora koji uključuje detektovanu oblast pretnje, a replaniranje putanje se mora obaviti u oblasti prozora, pre nego što letelica stigne na replanirani startni položaj.

U [169] je predložen metod za planiranje putanje mobilnih robota koji koristi jednostavni algoritam kolonije mrava (Simple Ant Colony Optimization Meta-Heuristic, SACO-MH). SACO-MH je modifikovan tako što donošenje odluka zavisi od rastojanja d između izvornog čvora i odredišnih čvorova, a mravi imaju i memoriju m gde smeštaju čvorove koje su posetili, što pomaže u rešavanju problema stagnacije. Novi algoritam, SACO-MHdm koristi fazi kriterijumsku funkciju radi evalucije najboljih putanja, a fazi sistem zaključivanja je izmenjen pomoću jednostavnog algoritma za

podešavanje (Simple Tuning Algorithm). Predloženo rešenje može da se koristi kao planer putanje u statičkom i dinamičkom okruženju, i ima dva moda rada: za virtuelna okruženja i za rad u realnim uslovima. Planiranje putanje mobilnih robota pomoću pojačanog (intensified) algoritma kolonije mrava je opisano u [170].

5.4 Algoritam svica

Adaptacija algoritma svica i njegova primena na rešavanje problema planiranja putanje UCAV letelica je predložena u [171]. Najpre je uveden matematički model radi transformacije problema planiranja putanje u D-dimenzionalni optimizacioni problem. Prostim matematičkim postupkom se vrši transformacija koordinatnog sistema u novi, tako da je horizontalna osa novog koordinatnog sistema duž koja povezuje početnu i krajnju tačku na putanji. Nova horizontalna X osa se zatim deli u D jednakih segmenata, a onda se vrši optimizacija vertikalne koordinate Y' na vertikalnoj liniji za svaki čvor kako bi se dobila grupa tačaka koje u stvari predstavljaju vertikalne koordinate D tačaka. Putanja se dobija prostim povezivanjem ovako dobijenih tačaka. Kao indikatori performansi uzeti su bezbednost i potrošnja goriva, odnosno, putanja treba da bude izabrana tako da stepen izloženosti pretnjama u toku leta bude na minimalnom nivou, i takođe da potrošnja goriva UCAV letelice bude najmanja moguća. Ukupni indikator performansi je prikazan sledećom jednačinom, gde je J_t indikator koji se odnosi na pretnje, J_f indikator koji se odnosi na potrošnju goriva, dok k predstavlja koeficijent za uspostavljanje ravnoteže između zahteva za minimizacijom pretnji i potrošnje goriva, i uzima vrednost iz intervala [0,1], tako da veća vrednost govori o tome da je prioritet bezbednost:

$$minJ = kJ_t + (1-k)J_f \tag{5.5}$$

Autori su izvršili modifikaciju originalnog algoritma svica sa ciljem ubrzanja konvergencije. Prva modifikacija se odnosi na dodavanje Levijevog leta sa veličinom koraka α. Ovo je učinjeno kako bi se podstakla eksploatacija u fazama kada su individue tj. svici blizu rešenja. Druga modifikacija je dodavanja razmene informacija između najboljih tj. najsjajnijih (top) svitaca. U modifikovanom algoritmu postoje dva nova parametra, koeficijent apsorpcije svetlosti i udeo svitaca koji čine grupu top svitaca.

Algoritam 5.2: Modifikovani algoritam svica za planiranje putanje UCAV letelica

begin
  Inicijalizacija. Postaviti brojač generacija $G = 1$. Na slučajan način inicijalizovati populaciju P od n svitaca tako da svaki svitac odgovara potencijalnom rešenju datog problema; definisati koeficijent apsorpcije svetlosti γ, postaviti parametar α koji kontroliše veličinu koraka i inicijalnu privlačnost β_0 za $\gamma = 0$;
  Generisanje rotacionog koordinatnog sistema. Transformisati originalni koordinatni sistem u novi rotacioni koordinatni sistem; konvertovati informacije o pretnjama na bojnom polju u rotacioni koordinatni sistem i podeliti novu osu X u D jednakih segmenata. Svako potencijano rešenje $P = \{p_1, p_2, \ldots, p_D\}$ je niz realnih brojeva koji predstavljaju koordinate od D;
  Evaluirati troškove pretnje/intenzitet svetlosti J za svakog svica iz P na osnovu jednačine (5.5);
  while *(nije ispunjen kriterijum prekida rada algoritma) ili* $(G < MaxGeneration)$ **do**
    Sortirati populaciju svitaca P od najboljeg do najgoreg prema troškovima pretnje/intenzitetu svetlosti J svakog svica;
    Razmeniti informacije između najboljih (top) svitaca;
    for $i = 1$ **to** n **do**
      for $j = 1$ **to** n **do**
        if $J_j < J_i$ **then**
          Pomeriti svica i prema svicu j;
        end
        Varirati privlačnost sa distancom r pomoću $exp[-\gamma r^2]$;
        Evaluirati nova rešenja i ažurirati troškove pretnje/intenzitet svetlosti;
      end
    end
    Evaluirati troškove pretnje/intenzitet svetlosti za svakog svica u P;
    Sortirati populaciju svitaca P od najboljeg do najgoreg prema troškovima pretnje/intenzitetu svetlosti J svakog svica;
    $G = G + 1$;
  end
  Izvršiti inverznu transformaciju koordinata u originalne i generisati izlaz;
end

Pokazalo se da su vrednosti 1.0 i 0.25 najpogodnije za ova dva pomenuta parametra.

Modifikovani algoritam svica koji autori predlažu za rešavanje problema planiranja putanje UCAV letelica je dat u Algoritmu 5.2. Intenzitet svetlosti svica i je predstavljen preko funkcije cilja date jednačinom (5.5), tako da je intenzitet svetlosti koju svitac emituje obrnuto proporcionalan troškovima pretnje. Simulacijom je pokazano da je modifikovani algoritam svica superioran ili kompetitivan sa originalnim algoritmom svica, a ima bolje performanse u poređenju sa ostalim analiziranim populacionim metodama optimizacije, i to: ACO, biogeografska optimizacija (biogeography-based optimization, BBO), DE, ES, GA, probabilističko inkrementalno učenje (probabilistic-based incremental learning, PBIL), PSO i stud genetski algoritam (SGA).

Modifikovani algoritam svica je korišćen i za rešavanje 3D problema planiranja putanje autonomnih podvodnih vozila [172]. Izmene algoritma se odnose na dizajn slučajnog koraka (random movement step) koji je podešen da bude jednak rastojanju između dva svica. Takođe, u cilju izbegavanja pogrešnog leta svica napravljena je autonomna strategija kojom se nova pozicija svica ocenjuje pre nego što se on pomeri, odnosno, svitac će promeniti svoju poziciju samo ako je ona korisna u kontektstu rešavanja problema. Uvedena su i dva operatora: operator isključivanja, koji se koristi radi boljeg izbegavanja prepreka; i operator povlačenja, da bi se povećala brzina konvergencije i glatkost putanje. Kriterijumska funkcija ima dva cilja: minimizaciju dužine putanje i bezbednost. Predloženi algoritam je putem simulacija upoređivan sa PSO, GA i klasičnim FA algoritmom, pri čemu je korišćeno 6 standardnih test funkcija: sphere, rosenbrock, rastrigin, grienwank, ackley i zakharo.

5.5 Kukavičje pretraživanje

Nova metoda za rešavanje problema planiranja putanje mobilnih robota u nepoznatom ili delimičnom poznatom dvodimenzionalnom okruženju u kojem se nalaze statičke prepreke opisana je u [173]. Algoritam je zasnovan na Levijevom letu i metaheurističkom algoritmu kukavičjeg pretraživanja. Robot treba da se kreće ka destinaciji izbegavajući prepreke koje mu stoje na putanji. Kada se robot nađe u blizini prepreka (senzori detektuju prepreku) aktivira se algoritam kukavičjeg pretraživanja kako bi se našla najbolja pozicija gnezda robota (u skladu sa definisanom ciljnom funkcijom) i time izbegla prepreka. Planiranje putanje mobilnog robota zavisi od dva tipa ponašanja koja on ispoljava: izbegavanje prepreka i dostizanje ciljne pozicije.

Kad je reč o izbegavanju prepreka, robot teži da rastojanje između položaja gnezda i prepreke bude maksimalno bezbedno. Ponašanje koje se tiče dostizanja ciljne pozicije se realizuje tako da pozicija najboljeg gnezda uvek bude na minimalnoj udaljenosti od ciljne pozicije. Ciljna funkcija koja treba da zadovolji ova dva zahteva se može predstaviti sledećom jednačinom:

$$F = C_1 \frac{1}{\underset{OB_j \in OB_d}{min} \|Dist_{N-OB_d}\|} + C_2 \|Dist_{N-G}\| \tag{5.6}$$

gde su OB_j prepreke iz okruženja koje su u senzorskom opsegu robota, $\|Dist_{N-G}\|$ je rastojanje između gnezda i cilja, $\|Dist_{N-OB_d}\|$ rastojanje između gnezda i prepreke, a C_1 i C_2 su kontrolni parametri. Ako je vrednost parametra C_1 velika robot će biti daleko od prepreka, a ako je mala on će biti blizu što povećava verovatnoću kolizije sa preprekama. Parametar C_2 utiče na dužinu putanje, što je veća vrednost ovog parametra putanja će biti kraća. Ispravno podešavanje ovih parametara utiče na konvergenciju i performanse algoritma, a vrednosti koje se koriste u simulacijama i ekperimentima su podešene na način kako se to radi u svim prirodom inspirisanim metaheurističkim algoritmima, primenom metoda pokušaja i greške (trial and error). Pseudokod predloženog algoritma je dat u Algoritmu 5.3.

Algoritam 5.3: Planiranje putanje mobilnih robota primenom kukavičjeg pretraživanja

1 Inicijalizacija startne i ciljne pozicije;
2 Kretati se prema cilju dok se ne naiđe na prepreku;
3 Kada se prepreka nalazi na putanji, primeniti algoritam kukavičjeg pretraživanja;
4 Generisati inicijalnu populaciju gnezda, gde svako gnezdo predstavlja jedno rešenje predloženog optimizacionog problema;
5 Izračunati tekuće najbolje gnezdo i naći globalno najbolje gnezdo;
6 Odbaciti najgore gnezdo i napraviti novo gnezdo primenom Levijevog leta;
7 Nastaviti kretanje robota ka poziciji najboljeg gnezda;
8 Ponavljati korake 2–7 dok robota ne savlada sve prepreke ili dok ne stigne do ciljne pozicije;

Efikasnost predloženog rešenja je proveravana simulacijama i ekperimentom u realnom vremenu. Za realizaciju eskperimenta su korišćene prepreke različite veličine i oblika, i Khepera-III mobilni robot.

5.6 Algoritam veštačke kolonije pčela

Primena modifikovanog algoritma kolonije pčela (ABC) u cilju optimizacije problema planiranja putanje u dvodimenzionalnom okruženju je data u [174]. Modifikacija originalnog ABC algoritma ogleda se u primeni balansirane evolutivne strategije (BE), što omogućuje algoritmu da u svakoj iteraciji koristi konvergencioni status kako se bi manipulisalo tačnošću eksploracije/eksploatacije i ostvario balans između lokalne eksploatacije i globalne eksploracije. Novi algoritam, nazvan BE-ABC, se razlikuje od originalnog ABC u tome što koristi parametar $trial(i)$ za podešavanje tačnosti eksploracije/eksploatacije i ima novu strategiju za generisanje pčela skauta. Dodatno, u jednačine koje određuju operacije ukrštanja i mutacije je dodat faktor konvergencije kako bi se podešavala tačnost eksploatacije, a broj elemenata koji učestvuju u pomenutim operacijama je adaptivan. Matematički opis problema je identičan onom u našem istraživanju u 6.1, a veoma slična je i organizacija simulacija. Rezultati simulacija su pokazali poboljšanje u odnosu na osnovni ABC algoritam i dve njegove modifikacije, I-ABC i IF-ABC.

ABC algoritam u kome je poboljšanje realizovano primenom teorije haosa je primenjen na problem planiranja putanje UCAV letelica u [175]. Modelovanje problema je urađeno kao u našem istraživanju u 6.1, a pretnje su predstavljene oblastima kružnog oblika tako da je ranjivost na pretnju predstavljenja verovatnoćom koja je obrnuto proporcionalna udaljenošću od centra oblasti pretnje. Kriterijumska funkcija uzima u obzir troškove pretnje i potrošnju goriva, i definisana je na identičan način kao u našem istraživanju u 6.1. Haos je nestabilnost determinističkih sistema u konačnom faznom prostoru koji često postoji u nelinearnim sistemima, takva da mala promena neke varijable može da izazove veliku promenu drugih parametara sistema. Primenjena modifikacija ABC teži da iskoristi ergodicitet i neregularnost haotične varijable u cilju nalaženja optimalnih parametara i izvlačenja iz lokalnog optimuma. Nakon faze pretraživanja svake pčele, primenjuje se haotično pretraživanje u okolini tekućeg najboljeg rešenja kako bi se izabralo bolje rešenje za sledeću generaciju.

5.7 Ostale prirodom inspirisane metaheuristike

Za rešavanje problema planiranja putanje UCAV u 3D dinamičkom okruženju koje podrazumeva pokretne zone opasnosti predlaže se u [176] predator-plen golubovima

inspirisan optimizacioni algoritam (predator-prey pigeon-inspired optimization, PP-PIO). Golubovima inspirisani algoritam (pigeon-inspired optimization, PIO) spada u grupu algoritama inteligencije rojeva i bazira se na kretanju golubova. Domaći golubovi nalaze put do kuće na osnovu tri alata: magnetnog polja, sunca i određenih obeležja predela (landmarks). U toku rada algoritma moguće je koristiti dva modela operatora: mapa i kompas model, koji se bazira na magnetnom polju i suncu; i landmark model, koji se bazira na obeležjima predela. Problem konvergencije koji je inherentan ovom algoritmu se ovde pokušava rešiti uvođenjem koncepta predator-plen. Predatori se uvode u populaciju sa ciljem uklanjanja najgorih golubova (individua) i da bi naterali ostale individue da se udalje od pomenutih rešenja. Funkcija troškova se dizajnira tako da uključi karakteristike putanje, ograničenja i zahteve koje treba zadovoljiti. U ovom radu ona je predstavljena kao zbir troškova dužine putanje, altitude, zona opasnosti, potrebne snage, kolizija i glatkoće. Matematički model putanje se konstruiše polazeći od početne i krajnje tačke koje se povezuju, a zatim se pravi projekcija date linije ST na XY ravan. Projekcija $S'T'$ se nakon toga deli u $D+1$ jednakih segmenata i određuje se vertikalna ravan na svakoj tački segmenta. Uzimaju se tačke u svakoj od ovih vertikalnih ravni i međusobno se povezuju kako bi se formirala putanja leta. Na ovaj način se problem planiranja putanje transformiše u problem nalaženja optimalnih vrednosti koordinata koje će dati najbolje vrednosti fitnes funkcije, odnosno, u optimizacioni problem. Putanja koja se ovim algoritmom generiše se sastoji od linijskih segmenata, što nije dobro rešenje za UCAV pa se primenjuje strategija k-trajektorija da bi se putanja učinila glatkom. Rezultati suimulacije su pokazali da je PPPIO efikasniji od osnovnog PIO, PSO i DE u rešavanju problema planiranja putanje UCAV u 3D okruženju.

Algoritam gravitacionog pretraživanja se koristi za optimalno planiranje putanje mobilnog robota u [177], odnosno UAV letelica u [178]. Za planiranje trajektorije UCAV letelica Duan i dr. [179] su predložili algoritam optimizacije inteligentnih vodenih kapljica (intelligent water drops optimization), a u [180] je predstavljena poboljšana verzija algoritma veštačkog jata riba (improved artificial fish swarm algorithm IAFSA), i njegova primena na rešavanje problema planiranja putanje mobilnog robota. Eksperimenti za ispitivanje performansi predloženog rešenja su izvršeni primenom robotskog operativnog sistema na Pioneer 3-DX mobilnom robotu.

5.8 Hibridni algoritmi

Rešenje za problem planiranja putanje UAV letelica na moru primenom hibridnog algoritma koji se bazira na diferencijalnoj evoluciji i algoritmu optimizacije rojevima čestica sa kvantnim ponašanjem (QPSO) je dato u [181]. Predloženi algoritam se sastoji iz dve faze: prvo se radi ažuriranje populacije primenom QPSO, a onda se primenjuje modifikovani DE algoritam. U kanoničkom DE, donorski vektor se generiše na osnovu tri individue iz tekuće populacije, dok se generisanje donorskog vektora u modifikovnom DE obavlja tako što se na globalno najbolju poziciju dobijenu primenom QPSO u prvoj fazi dodaje težinska razlika dva ili više slučajno izabranih personalnih najboljih pozicija. Takođe, kao i kod kanoničkog DE, nakon primenjene operacije mutacije obavlja se modifikovana operacija ukrštanja. Pre nego što se konstruiše putanja autori predlažu jednostavan metod za prethodno tretiranje (pretraitment) terena tj. okruženja, koji se obavlja u dve faze. U prvoj fazi se nalaze konture svakog ostrva, što podrazumeva nalaženje graničnih tačaka objekta. Podešavanje tačaka dobijenih u prvoj fazi prema elipsi se obavlja u drugoj fazi. Autori su usvojili pretpostavku da UAV leti na konstantnoj nadmorskoj visini tako da se problem svodi na 2D pretraživanje smanjujući na taj način potrebno vreme izračunavanja i neophodnu memoriju. Putanja se deli u n tačaka, i data je nizom on n članova pri čemu svaki član predstavlja koordinate date tačke. Dakle, svaka čestica koja predstavlja rešenje problema u vidu potencijalne putanje ima dimenziju $2n$. Autori koriste B-Spline krivu za konstruisanje putanje. S obzirom da je usvojena pretpostavka o konstantoj visini leta, analiziraju se samo troškovi putanje koji se odnose na dužinu putanje, izloženost pretnjama i ograničenja ugla okretanja. Izvršena je komparativna analiza performansi predloženog algoritma i nekoliko prirodom inspirisanih metoda kao što su genetski algoritam, diferenijalna evolucija, standardni PSO, hibridni PSO sa diferencijalnom evolucijom, i QPSO. Eksperimentalni rezultati su pokazali superiornost predloženog algoritma u pogledu robustnosti, kvaliteta rešenja i konvergencije.

Hibridni algoritam koji se zasniva na nelinearnom vremenski promenljivom PSO (nonlinear timevarying PSO, NTVPSO) i modifikovanom, samoadaptivnom DE algoritmu čiji je operator mutacije baziran na rangiranju (ranking based self-adaptive DE, RBSADE) je razvijen za potrebe planiranja putanje mobilnih robota u 2D okruženju [182]. Algoritam radi tako što se nakon svake iteracije primenjuje modifikovani DE

algoritam radi efikasnije evolucije personalnih najboljih pozicija čestica, odnosno, da bi se čestice izvukle iz stagnacije. Samoadaptivni mehanizam predloženog algoritma služi za podešavanje tri kontrolna parametra: težina inercije, kognitivno i socijalno ubrzanje. Matematičko modelovanje radnog okruženja i putanje je obavljeno na način koji se često koristi u literaturi, a koji smo i mi koristili u našem istraživanju u 6.1. Prepreke se u ovom radu predstavljaju mnogouglovima, a ciljevi optimizacije su minimizacija troškova koji se odnose na sigurnost putanje i njenu dužinu. Sigurnost putanje se defniše kao suma minimuma distanci između svakog segmenta putanje i najbliže prepreke. Posmatrani problem spada u optimizacioni problem sa ograničenjima, jer je cilj naći putanju koja nema kolizija sa preprekama. Nakon izračunavanja stepena narušavanja ograničenja, koji je jednak odnosu broja narušavanja ograničenja i ukupnog broja prepreka, primenjuje se pravilo za biranje najbolje putanje od više mogućih. Pravilo se primenjuje na sledeći način: (1) ako dve putanje imaju isti stepen narušavanja ograničenja, bira se ona putanja koja ima bolju fitnes vrednost; (2) ako dve putanje imaju različit stepen narušavanja ograničenja, bira se ona putanja koji ima manju vrednost datog stepena. Rezultati simulacije su pokazali da je predloženo rešenje efikasnije u poređenju sa nekoliko popularnih prirodom inpirisanih metoda: JADE, vremenski promenljiva PSO, gravitaciona pretraga, i modifikovani GA algoritam.

Mo i Xu su predložili hibridni algoritam za rešavanje problema planiranja putanje robota u statičkom okruženju, koji se zasniva na kombinaciji biogeografske optimizacije (Biogeography-based Optimization, BBO) i standardnog PSO algoritma [183]. U BBO algoritmu se za predstavljanje rešenja koriste ostrva, pri čemu se osobine ostrva u smislu pogodnosti za naseljavanje (habitability) predstavljaju skupom promenljivih koje se nazivaju suitability index variables (SIV). Potencijalna rešenja se ocenjuju preko odgovarajućeg indeksa pogodnosti za naseljavanje (habitat suitability index, HSI), pri čemu veća vrednost ovog parametra označava ostrvo koje je pogodnije za naseljavanje, tj. predstavlja bolje rešenje optimizacionog problema. Bolja rešenja trpe manje promene u toku rada algoritma od lošijih, ali i dele svoje osobine sa lošijim rešenjima. BBO algoritam ima migracioni operator koji funkcioniše na sledeći način: kada se ostrvo izabere radi modifikcije, posmatra se njegov stepen imigracije da bi se na probabilistički način utvrdilo koje promenljive iz SIV treba modifikovati; na

osnovu stepena emigracije drugih ostrva se na probabilistički način određuje koja ostrva treba da migriraju slučajno izabranu SIV vrednost ka posmatranom rešenju tj. ostrvu koje se modifikuje. Nakon migracije obavlja se i proces mutacije kojim se obezbeđuje diverzitet u populaciji. Za modelovanje putanje koristi se približna Voronoi mreža ograničavanja (approximate Voronoi boundary network, AVBN) pri čemu se koriste generalizovani Voronoi dijagrami u rasterski baziranom prostoru. Okruženje u kome se robot kreće se posmatra kao mreža (grid) dimenzija 400x400, a svaki element grida je jedna mala oblast iz realnog okruženja. Prisustvo prepreke u elementu grida se obeležava celobrojnim vrednostima, gde vrednost 0 označava prostor bez prepreka. Uloga PSO u hibridnom algoritmu je da vrši ažuriranje položaja čestica kojim se povećava diverzitet populacije u BBO algoritmu. U osnovnom BBO, ako nijedno ostrvo nije odabrano za imigraciju, onda se neće promeniti ni odabrana ostrva za emigraciju, što negativno utiče na diverzitet populacije. U modifikovanom algoritmu, kada se desi ovaj slučaj, primenjuje se PSO i njegova strategija ažuriranja čestica kako bi se modifikovala ostrva koja nisu izabrana za imigraciju.

Novi pristup za određivanje optimalne putanje u okruženju koje sadrži više robota, a koji se bazira na kombinovanju poboljšanog PSO algoritma (improved particle swarm optimization, IPSO) i algoritma diferencijalno perturbovane brzine (differentially perturbed velocity, DV) je dat u [184]. Cilj algoritma je da se minimizuje dužina putanje, odnosno vreme stizanja na destinaciju za sve posmatrane robote, uz ograničenje da roboti ne smeju međusobno da se sudaraju (dinamičke prepreke) niti da imaju kolizije sa preprekama prisutnim u okruženju (statičke prepreke). Dodatno, optimalne putanje treba da budu takve da minimizuju potrošnju energije koja se posmatra kao broj okretanja robota potrebnih da bi stigao na destinaciju. IPSO algoritam je dobijen modifikacijom osnovnog PSO tako što su primenjeni adaptivno podešavanje težina i koeficijenti akceleracije koji imaju pozitivan efekat na brzinu konvergencije algoritma. Dodatno, IPSO-DV koristi diferencijalni operator iz DE algoritma u jednačini za ažuriranje brzina IPSO algoritma. On se primenjuje na odgovarajućim pozicijama vektora susednih čestica koje nisu njihove najbolje pozicije. Ukupna ciljna funkcija je data sa:

$$F = \lambda_1 F_1 + \lambda_2 F_2 + \lambda_3 F_3 + \lambda_4 F_4 \tag{5.7}$$

gde je: F_1 – deo ciljne funkcije koji se odnosi na najkraću putanju; F_2 – funkcija odbija-

nja (repulsive function), definiše se kao relativno rastojanje između robota i prepreka; F_3 – predikcija dinamičkog položaja prepreka u okruženju; F_4 – deo ciljne funkcije koji se odnosi na glatkost putanje, koja se definiše kao ugao između hipotetičkih linija koje povezuju ciljnu tačku i dve susedne najbolje pozicije robota u svakoj iteraciji. Koeficijenti λ_i, koji predstavljaju doprinos svake od delova ciljne funkcije se podešavaju u toku simulacije. Efektivnost predloženog algoritma je proveravana putem simulacije u okruženju od 12 soft botova kružnog oblika i sedam prepreka različitog oblika. Takođe, vršen je i realan ekperiment sa Khepera-II robotom, minijaturnim robotom prečnika 7 cm. Okruženje se sastojalo od dva robota i osam statičkih prepreka. U delu koji se odnosi na simulaciju, predloženi algoritam je upoređivan sa DE i IPSO, a u ekperimentalnom delu sa PSO, gravitacionim pretraživanjem, IPSO i DE. Rezultati su utvrdili da su performanse predloženog rešenja značajno bolje od konkurenata.

Wang i dr. su napravili algoritam za planiranje putanje bespilotnih borbenih letelica korišćenjem algoritma slepog miša sa mutacijama (bat algorithm with mutation, BAM) [185]. Operator mutacije iz DE algoritma se primenjuje tako da doprinosi ubrzanju konvergencije originalnog BA algoritma uz očuvanje njegove robustnosti. Razlozi za primenu hibridnog operatora mutacije su poboljšanje loših rešenja i poboljšanje sposobnosti eksploracije novog algoritma. Mutacioni operator iz DE se koristi tako da poboljša originalni algoritam slepog miša generišući novo rešenje za svakog slepog miša sa verovatnoćom $1 - r$ koristeći slučajni hod. Matematički model problema koji se ovde primenjuje je veoma poznat i često korišćen, i identičan je modelu kojeg smo koristili u našem istraživanju u 6.1. Autori koriste dva indikatora performansi: indikator bezbednosti i indikator potrošnje goriva. Izračunavanje ovih indikatora i definisanje kriterijumske funkcije se obavlja na isti način kao i u našem istraživanju. Pseudokod algoritma je dat u Algoritmu 5.4. Algoritam je pokazao superiornost u odnosu na nekoliko poznatih metaheuristika: ACO, BBO, DE, ES, GA, PBIL, PSO, i SGA, pri čemu je poređenje vršeno za različite maksimalne generacije i različite dimenzije.

Hibridni algoritam nazvan genetsko kaljenje (genetic annealing), koji predstavlja kombinaciju genetskog algoritma i simuliranog kaljenja predložen je u [186] za planiranje putanje više robota u dinamičkom okruženju. Hibridni pristup rešava problem preuranjene konvergencije genetskog algoritma, i brže konvergira od genetskog algoritma i simuliranog kaljenja.

Algoritam 5.4: BAM algoritam za planiranje putanje UCAV

begin
  Inicijalizacija;
  Generisanje rotacionog koordinatnog sistema. Transformacija originalnih koordinata; konvertovati informacije o bojnom polju u rotacioni koordinatni sistem i podeliti osu X u D jednakih delova. Svako dopustivo rešenje je niz od D koordinata koje su realni brojevi.;
  Evaluiranje troškova pretnje za svakog slepog miša iz populacije P;
  while *(kriterijum zaustavljanja nije ispunjen) or* $(t < MaxGeneration)$ **do**
    Sortiranje populacije slepih miševa P od najboljeg do najgoreg na osnovu troškova pretnje;
    for $i = 1$ **to** NP **do**
      Izabrati na slučajan način uniformno $r_1 \neq r_2 \neq r_3 \neq i$;
      $r_4 = \lceil NP * rand \rceil$;
      $v_i^t = v_i^{t-1} + (v_i^t - x_*) * Q$;
      $x_i^t = x_i^{t-1} + v_i^t$;
      if $rand > r$ **then**
        $x_u^t = x_* + \alpha\epsilon^t$
      else
        $x_u^t = x_{r1}^t + F(x_{r2}^t - x_{r3}^t)$
      end
      Evaluiranje fitnes funkcije za potomke x_u^t, x_i^t, x_{r4}^t;
      Selektovanje potomka x_k^t sa najboljom fitnes vrednošću među potomcima x_u^t, x_i^t, x_{r4}^t;
      if $rand < A$ **then**
        $x_{r4}^t = x_k^t$
      end
    end
    Evaluiranje troškova pretnje za svakog slepog miša iz populacije P;
    Sortiranje populacije slepih miševa P od najboljeg do najgoreg na osnovu troškova pretnje;
    $t = t + 1$;
  end
  Inverzna transformacija koordinata finalne optimalne putanje u originalni oblik, i generisanje izlaza;
end

Trodimenzionalni planer za UCAV realizovan pomoću hibridnog metaheurističkog algorirtma koji koristi ACO i DE, pri čemu se DE koristi za optimizovanje tragova

feromona u toku procesa ažuriranja feromona poboljšanog ACO algoritma je opisan u [187]. Ciljevi algoritma su minimizacija potrošnje goriva i maksimalna sigurnost. Nakon generisanja optimalnih putanja primenjuje se metod k-trajektorija za postizanje glatkosti. Hibridni algoritam za planiranje putanje mobilnih robota u statičkom okruženju koji se sastoji od ACO i poboljšanog GA opisan je u [188]. Predloženi algoritam funkcioniše tako što se prvo primenjuje ACO algoritam kako bi se našle suboptimalne putanje bez kolizija, koje služe kao inicijalna populacija za GA algoritam. Za razliku od originalnog GA algoritma, predloženo poboljšanje sadrži operator brisanja koji se zasniva na domensko heurističkom znanju, tako da odgovara optimalnom planiranju putanje za mobilne robote. Predloženi algoritam se prvo primenjuje na jednociljnom problemu planiranja putanje, gde se posmatra samo dužina putanje, a zatim se definiše i rešava višeciljni problem planiranja putanje, pri čemu su ciljevi: dužina putanje, bezbednost i glatkost. U [189] se predlaže višeciljni hibridni algoritam zasnovan na gravitacionom pretraživanju i PSO algoritmu. Algoritam koristi dve kriterijumske funkcije za generisanje optimalnih putanja u statičkom okruženju u kojem se nalaze izvori opasnosti i prepreke.

Ju i dr. [190] su predložili hibridni algoritam koji kombinuje evolutivni pristup i PSO. Kodiranje na bazi binarnog stabla se koristi za skalabilnu reprezentaciju putanje, a algoritam dodaje glupe čvorove (dummy nodes) u stablo da bi se rešili problemi kombinacije dva algoritma. Algoritam radi tako što se prvo kreiraju inicijalne putanje i odgovarajuća binarna stabla. Svaka putanja se zatim evaluira na bazi njene fitnes vrednosti, i putanje se sortiraju od najbolje do najgore. Prva polovina elitnih putanja se zadržava, a ostale se brišu. Nakon toga se elitne putanje poboljšavaju primenom PSO algoritma, a nove putanje se kreiraju putem operatora ukrštanja pri čemu su roditelji iz grupe elitnih putanja. Kombinacija kukavičjeg pretraživanja i diferencijalne evolucije je predložena za rešavanje planiranja putanje UAV u 3D okruženju [191]. Algoritam ima dva cilja: minimizaciju potrošnje goriva i izbegavanje pretnji, a prostor se modeluje gridom. Kukavičjim pretraživanje se traži optimalna putanja, a operatori mutacije i ukrštanja iz DE algoritma se koriste umesto originalnog metoda za selekciju kukavice iz CS metode. Na ovaj način, integracija DE u CS unosi diverzitet u populaciju i poboljšava efikasnost pretraživanja.

6 PLANIRANJE PUTANJE ROBOTA PRIMENOM BRAIN STORM ALGORITMA

6.1 Planiranje putanje UCAV letelica primenom brain storm algoritma

Brain storm optimizacioni algoritam je primenjivan u [192] za rešavanje problema formacijskog leta više UAV letelica. Predloženo rešenje koristi modifikovani BSO algoritam i nelinearni mod kontrole odstupajućeg horizonta (receding horizon control) UAV da bi se odredili RHC kontrolni parametri za UAV formacijski let. RHC je kontrolni metod baziran na optimizaciji koji se koristi za formacijsko letenje i druge kooperativne zadatke. Osnovna ideja RHC je online optimizacija pomeranja i odstupanja (recede/move) kojom se nakon prvog unosa optimalne komande sekvence primenjuje postupak iterativnog rešavanja optimalnog kontrolnog problema i ažuriranja stanja. Primenom ovog pristupa se globalni kontrolni problem sa ograničenjima efikasno deli na manje lokalne optimizacione probleme čime se smanjuje računarska kompleksnost cele procedure. U modifikovanom BSO se koristi novi metod za klasterovanje koji koristi sortiranje fitnes vrednosti i dva moguća rešenja za realizaciju zamene starog rešenja boljim: probabilistički pristup i primena teorije haosa. Osim toga, centri klastera dobijaju važniju ulogu vođenja ostalih individua.

UAV letelice opremljene optičkim senzorima, koji imaju mogućnost ograničenog okretanja tako da pokriju deo prostora, mogu da se koriste za izviđanje i nadgledanje, odnosno pretraživanje jedne ili više meta u nekom 2D regionu. Problem pretraživanja može da se formuliše kao višeciljni složeni optimizacioni problem sa ograničenjima, u kojem se određuje položaj UAV letelice i optičkog senzora tako da troškovi budu minimalni. U [193] se za rešavanje ovog problema predlaže pristup pretraživanju na bazi razdvajajućeg odstupajućeg horizonta (decoupling receding horizon search approach) i BSO algoritma.

U ovom poglavlju mi predlažemo rešenje za problem planiranja putanje UCAV letelica bazirano na brain storm optimizacionom algoritmu. Kvalitet predloženog rešenja je testiran poređenjem dobijenih rezultata sa ostalim najpoznatijim metaheuristikama.

UCAV se često koriste u neprijateljskim okruženjima, te su kontrolni sistemi za

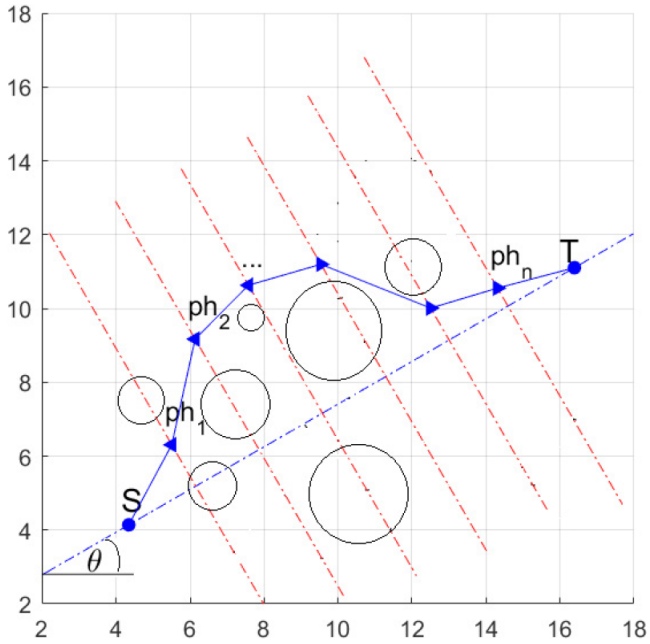
njih veoma zahtevni. Problem planiranja putanje borbenih bespilotnih letelica je jedan od najvažnijih delova autonomnog kontrolnog modela za UCAV. On se odnosi na problem nalaženja optimalne putanje od početne tačke do konačnog odredišta, uzimajući u obzir nekoliko ciljeva i ograničenja. Cilj planiranja putanje UCAV letelice je nalaženje tj. izračunavanje suboptimalne rute leta tako da letelica završi svoj let za odgovarajuće vreme, izbegne sve moguće pretnje, preživi i uspešno kompletira misiju.

6.1.1 Matematički model

Planiranje putanje borbenih bespilotnih letelica predstavlja aktivnu temu istraživanja i predložene su razne metode modelovanja putanje. Problem planiranja putanje UCAV se može opisati kao optimizacioni problem gde je cilj da se nađe optimalna ruta polazeći od početke tačke (S) do cilja (T), u skladu sa određenim metrikama. Optimalna putanja se može definisati na različite načine, s obzirom da se mogu razmatrati različite osobine kao što su najkraća putanja, putanja sa najvećom glatkošću, minimalna potrošnja goriva, najbezbednija putanja i sl. U ovom istraživanju se koriste dva različita kriterijuma: potrošnja goriva i stepen sigurnosti. Borbene bespilotne letelice se kreću u trodimenzionalnom prostoru, ali u našem istraživanju nije razmatrana nadmorska visina (altituda), što znači da smo za opis problema planiranja putanje koristili dvodimenzionalni prostor. Ovo pojednostavljenje je primenjeno i u drugim radovima iz literature [171], [185]. Za modelovanje rešenja smo usvojili metodu koja je uobičajena, ne samo za planiranje putanje UCAV [147], [174], nego i za planiranje putanje robota [148]. Korišćeni metod je jednostavan, ali efikasan. Osnovna ideja je da se obezbedi kretanje UCAV ka ciljnoj poziciji. Model putanje koji smo usvojili transformiše problem planiranja putanje u D-dimenzionalni problem na sledeći način. Prvi korak je da se tranformiše koordinatni sistem tako da je horizontalna osa (x-osa) linija koja povezuje početnu i ciljnu poziciju. Na ovaj način, startna pozicija postaje tačka $(0, 0)$, što je postignuto sledećim transformacijama:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} x_s \\ y_s \end{bmatrix} \tag{6.1}$$

gde (x, y) predstavljaju originalne koordinate, (x', y') su nove koordinate u transformisanom koordinatnom sistemu, (x_s, y_s) su koordinate startne pozicije, a θ je ugao od



Slika 6.1 Modelovanje putanje

x ose ka vektoru $\overrightarrow{ST}$:

$$\theta = \arcsin \frac{|y_T - y_S|}{\|\overrightarrow{ST}\|} \tag{6.2}$$

Nadalje, putanja se modeluje tako što se horizontalna osa, linija S-T, podeli putem n tačaka u $n+1$ jednakih segmenata. X-koordinate svake tačke su određene i fiksne kao što je prikazano na Slici 6.1. Po jedna tačka putanje se nalazi na svakoj crvenoj liniji prikazanoj na Slici 6.1. Y-koordinata svake tačke treba da bude pronađena za svaku optimalnu putanju. Na Slici 6.1 tačke putanje su označene sa $ph_1, ph_2, \ldots, ph_n$, gde n predstavlja dimenzionalnost optimizacionog problema, tj. $D = n$. Kompletna putanja UCAV se može predstaviti na sledeći način:

$$PH = (S, ph_1, ph_2, \ldots, ph_n, T) \tag{6.3}$$

6.1.2 Kriterijumi performansi

Cilj metoda za planiranje putanje je da se nađe optimalna ruta za UCAV pri čemu kriterijumi optimalnosti mogu biti različiti. Mi smo koristili dva kriterijuma: potrošnju goriva i stepen sigurnosti.

Potrošnja goriva je proporcionalna dužini putanje tako da može biti predstavljena kao funkcija dužine. Ukupna potrošnja goriva je definisana na sledeći način:

$$F_{fuel} = \int_0^L w_f dl \tag{6.4}$$

gde je L kriva putanje, a w_f predstavlja potrošnju goriva za svaku tačku putanje. Usvojili smo da je $w_f = 1$, što znači da je ukupna potrošnja goriva jednaki dužini putanje koja se računa kao zbir rastojanja između susednih tačaka na putanji. Ako startnu poziciju S obeležimo sa ph_0, a ciljnu poziciju T sa ph_{n+1}, dužina putanje se može izračunati na sledeći način:

$$L(PH) = \sum_{i=0}^{n} d(ph_i, ph_{i+1}) \tag{6.5}$$

gde je $d(a, b)$ euklidsko rastojanje između a i b.

Drugi kriterijum koji smo razmatrali je sigurnost. UCAV letelice se u toku leta mogu suočavati sa određenim pretnjama kao što su radari, radijacija, projektili, protivazdušna artiljerija i sl. Ove pretnje se mogu predstaviti kao kružne oblasti sa različitim poluprečnicima i stepenom pretnje ili težinom [171]. Ako putanja prolazi kroz oblasti koje sadrže određene opasnosti, verovatnoća da će UCAV letelica biti oštećena ili uništena je veća. S druge strane, verovatnoća oštećenja je jednaka nuli ako je putanja izvan kruga pretnje ili opasnosti. Slično kao u slučaju potrošnje goriva, troškovi pretnje se mogu izračunati kao:

$$F_{threat} = \int_0^L w_t dl \tag{6.6}$$

gde je L kriva putanje, a w_t predstavlja težinu pretnje za svaku tačku putanje. Da bi se odredili ukupni troškovi pretnje neophodno je uključiti sve pretnje. Svaka pretnja ima kružni opseg tj. oblast unutar koje je stepen pretnje konstantan. Ako je N_t broj pretnji sa kojima se UCAV letelica suočava u toku leta po segmentu putanje L_{ij},

ukupni troškovi pretnje se mogu izračunati pomoću sledeće jednačine [171]:

$$w_{t,L_{ij}} = \int_0^L \sum_{k=1}^{N_t} \frac{t_k}{((x-x_k)^2+(y-y_k)^2)^2} dl \tag{6.7}$$

gde t_k predstavlja stepen pretnje k čiji je centar (x_k, y_k). Da bi se pojednostavilo izračunavanje stepena pretnje ili sigurnosti, jednačina (6.7) može biti transformisana u diskretni model. Svaki segment putanje se može podeliti u $m+1$ jednakih podsegmenata pomoću m tačaka, tako da se sada stepen pretnje segmenta putanje L_{ij} može predstaviti kao prosečna vrednost stepena opasnosti u ovih m tačaka. U našem istraživanju smo koristili model prezentovan u radovima [171], [174] i [185], gde je svaki segment putanje podeljen sa 5 tačaka, tako da se troškovi pretnje koju proizvodi N_t pretnji u toku leta UCAV letelice po segmentu putanje L_{ij} mogu izračunati na sledeći način:

$$w_{t,L_{ij}} = \frac{L_{ij}^5}{5} \sum_{k=1}^{N_t} t_k \left(\frac{1}{d_{1,k}^4} + \frac{1}{d_{2,k}^4} + \frac{1}{d_{3,k}^4} + \frac{1}{d_{4,k}^4} + \frac{1}{d_{5,k}^4}\right) \tag{6.8}$$

gde L_{ij} predstavlja dužinu segmenta putanje između ph_i i ph_j, $d_{m,k}$ je euklidsko rastojanje između m-te podsegmentne tačke segmenta L_{ij} i pretnje k, dok t_k predstavlja nivo pretnje k-te pretnje.

U ovom poglavlju mi predlažemo BSO algoritam za rešavanje problema planiranja putanje borbenih bespilotnih letelica. U toku leta, UCAV letelica treba da izbegava oblasti gde se nalaze potencijalne pretnje uz minimizaciju potrošnje goriva. Minimizacijom potrošnje goriva uvećava se stepen pretnje, i obrnuto. U cilju izbegavanja oblasti pretnje produžava se dužina putanje što rezultuje u većoj potrošnji goriva. S obzirom da ova dva cilja ne možemo istovremeno da zadovoljimo, koristimo parametar λ koji određuje uticaj svakog od njih. Konačna funkcija cilja koja treba da bude minimizovana brain storm optimizacionim algoritmom je data sledećom jednačinom:

$$F(PH) = \lambda F_{fuel} + (1-\lambda) F_{threat} \tag{6.9}$$

6.1.3 Rezultati simulacije

Naš predloženi metod za planiranje putanje UCAV letelica je implementiran u programskom paketu Matlab R2016a, i sve simulacije su izvršene na računarskoj platformi sledećih karakteristika: Intel Core i7-3770K, CPU 4GHz, 8GB RAM, Windows 10 Professional OS. Da bismo testirali kvalitet metoda kojeg smo predložili, izvršili smo njegovo poređenje sa ostalim metodama predloženim u [171] i [185], i kvalitativno poređenje sa metodom predloženom u [194]. U [171] modifikovani algoritam svica (modified firefly algorithm, MFA) je korišćen za rešavanje problema planiranja putanje borbene bespilotne letelice. Modifikacije se odnose na dodavanje Levijevog leta sa dinamičkim parametrima za generisanje novih pozicija svitaca, i dodavanje razmene informacija između svitaca sa najboljim vrednostima fitnes funkcije. Ove modifikacije su povećale brzine lokalnog pretraživanja i konvergencije. U [185] je predložen algoritam slepog miša sa mutacijama (bat algorithm with mutation, BAM) za rešavanje istog problema koji se razmatra i u ovoj disertaciji. Operator mutacije diferencijalne evolucije je dodat u originalni algoritam slepog miša sa ciljem povećanja brzine konvergencije algoritma. Osim operatora mutacije, izvršena je još jedna modifikacija, i to podešavanje dinamičkih parametara algoritma slepog miša.

Simulacije smo organizovali na isti način kao što je to urađeno u [171] i [185], gde je razmatrano jedno okruženje leta. Startna pozicija je postavljena na (10, 10), dok su koordinatne ciljne tačke (55, 100). Parametar fitnes funkcije λ je postavljen na vrednost 0.5. Poznato je da u polju postoji pet zona pretnji. Sve informacije, tj. koordinate, radijusi pretnji i nivoi pretnji su date u Tabeli 6.1. U [171] i [185] predložene

Tabela 6.1 Informacije o pretnjama u polju

Broj	Lokacija(km)	Radijus pretnje(km)	Nivo pretnje
1	(45,50)	10	2
2	(12,40)	10	10
3	(32,68)	8	1
4	(36,26)	12	2
5	(55,80)	9	3

metode su upoređivane sa brojnim prirodom insipirisanim algoritmima, i to: genetski

algoritam (GA), stud genetski algoritam (stud genetic algorithm, SGA), diferencijalna evolucija (DE), evolutivna strategija (evolutionary strategy, ES), optimizacija rojevima čestica (PSO), algoritam kolonije mrava (ACO), originalni algoritam slepog miša (BA), i originalni algoritam svica (FA). U ovom istraživanju smo uključili te rezultate koje obuhvataju analizu performansi u dva različita slučaja: kada se menja maksimalni broj generacija i kada se razmatra različita dimenzionalnost problema. Dobijene rezultate smo normalizovali na isti način kao što je to urađeno u [171] i [185]. Dobijene vrednosti fitnes funkcije su smanjene za 50, što znači da ako je prijavljeni rezultat 2.9036, stvarna dobijena vrednosti fitnes funkcije je bila 52.9036.

Parametri za brain storm optimizacioni algoritam su podešeni nakon obavljanja preliminarnih računarskih eksperimenata. Verovatnoća generisanja novog slučajnog rešenja p_{5a} je postavljena na vrednost 0.2. Parametar za selektovanje jednog klastera p_{6b} ima vrednost 0.8. U slučaju da je izabran jedan klaster, verovatnoća da se koristi njegov centar p_{6bii} je 0.4. Verovatnoća da se radi kombinovanje centara klastera p_{6c}, je takođe postavljena na vrednost 0.4. Vrednost parametra p_{6bi} predstavlja verovatnoću da će neki klaster biti izabran i ona je proporcionalna broju individua u tom klasteru.

Prvi set simulacija uključuje promenu maksimalnog broja evaluacija fitnes funkcije. Za ove eksperimente, podešeno je da dimenzija bude 20, a maksimalni broj evaluacija fitnes funkcije je bio 1500, 3000, 4500, 6000 i 7500, isto kao u [171] i [185]. Veličina populacije za BSO je bila 20. Poređenje rezultata dobijenih primenom našeg metoda i rezultata dobijenih metodama opisanim u [171] i [185] dato je u Tabelama 6.2, 6.3 i 6.4. Tabela 6.2 sadrži poređenje za slučaj najboljih vrednosti fitnes funkcije, Tabela 6.3 poređenje za slučaj najgorih vrednosti fitnes funkcije, dok je poređenje prosečnih vrednosti fitnes funkcije dato u Tabeli 6.4.

Tabela 6.2 Normalizovane najbolje vrednosti fitnes funkcije dobijene u 100 izvršavanja za različite maksimalne brojeve generacija

Br.eval.	GA	SGA	DE	ES	PSO	ACO	FA	MFA	BA	BAM	BSO
1500	1.2604	1.7370	2.4179	9.6276	2.7827	10.7202	1.4713	0.7030	4.0662	0.6208	**0.6064**
3000	1.5073	1.3218	0.8503	10.6318	2.3469	10.8912	0.6577	0.5382	4.7582	**0.4900**	0.5314
4500	1.0991	1.1559	0.5319	11.1469	2.3738	9.9096	0.5459	0.4857	4.1112	0.4724	**0.4112**
6000	1.0792	0.7595	0.5047	11.2403	3.4276	12.3080	0.4931	0.4661	3.1463	0.4590	**0.4541**
7500	1.0640	1.0166	0.4792	12.3745	2.5221	7.1358	0.4753	0.4508	4.4072	0.4636	**0.3744**

Naš predloženi algoritam je našao najbolje rešenje u svim slučajevima, osim kada

je broj evalucija fitnes funkcije jednak 3000. Algoritam slepog miša sa mutacijama je našao bolju putanju UCAV letelice u tom slučaju, ne samo u odnosu na naš algoritam, nego i u poređenju sa svim ostalim algoritmima. Iz Tabele 6.2 se može uočiti da su dobri rezultati takođe postignuti primenom MFA, FA i DE algoritama, ali u slučaju kada broj evaluacija fitnes funkcije ima veće vrednosti. Ostali algoritmi su dali znatno slabije rezultate. Sa porastom broja evaluacija fitnes funkcije naš predloženi algoritam pokazuje konstantna poboljšanja. Veoma je važno uočiti da u slučaju kada broj evalucija fitnes funkcije ima vrednost 7500, svi algoritmi ispoljavaju osobinu stagnacije, dok naš predloženi BSO metod beleži poboljšanje.

Tabela 6.3 Normalizovane najgore vrednosti fitnes funkcije dobijene u 100 izvršavanja za različite maksimalne brojeve generacija

Br.eval.	GA	SGA	DE	ES	PSO	ACO	FA	MFA	BA	BAM	BSO
1500	10.2501	13.0102	25.3999	41.9676	28.6115	18.7099	28.0425	**4.6726**	39.0832	11.7494	8.7150
3000	8.2047	11.0529	18.6288	38.5875	25.7065	18.4316	29.3022	4.5749	29.9962	9.7666	**3.9941**
4500	10.5257	13.3517	13.8150	46.0828	29.6341	17.4223	27.8480	4.9631	31.1293	7.6952	**2.9817**
6000	6.7466	7.5385	10.4226	31.3944	33.0709	17.2147	26.5768	9.1502	24.9732	6.7334	**2.1708**
7500	8.9162	13.5830	8.9560	34.8908	27.3858	16.9896	26.3005	3.6783	24.7175	**3.3564**	3.4103

Na osnovu rezultata za slučaj najgorih vrednosti fitnes funkcija u 100 izvršavanja, prikazanih u Tabeli 6.3, može se zaključiti da je naš predloženi metod pronašao relativno dobra rešenja u svim slučajevima. Najgore rešenje je u slučaju kada je broj evalucija jednak 1500 i njegova vrednost je 8.7150, ali kada se broj evalucija fitnes funkcije povećao na 3000, najgore rešenje se značajno poboljšalo i imalo je vrednost 3.9941. Svi ostali algoritmi, sa izuzetkom MFA u slučaju kada je broj evaluacija 1500 i BAM u slučaju kada je broj evaluacija 7500, imaju veću razliku između najboljeg i najgoreg rešenja, što dokazuje robustnost našeg predloženog BSO metoda.

Prosečne vrednosti fitnes funkcije dobijene u 100 izvršavanja su prikazane u Tabeli 6.4. Naš predloženi algoritam je imao najbolje prosečne vrednosti za slučaj kada je broj evalucija fitnes funkcije 4500, 6000 i 7500, dok je BAM algoritam dao bolje rezultate kada je broj evalucija 1500 i 3000. BSO metodu je trebao veći broj generacija da bi našao optimalno rešenje, tj. imao je sporiju konvergenciju ali odličnu sposobnost nalaženja boljeg rešenja.

U [171] i [185] nisu prikazane standardne devijacije koji bi nam otkrile stvarnu robustnost analiziranih algoritama. Bez tih podataka možemo samo doneti grub i

Tabela 6.4 Normalizovane prosečne vrednosti fitnes funkcije dobijene u 100 izvršavanja za različite maksimalne brojeve generacija

Br.eval.	GA	SGA	DE	ES	PSO	ACO	FA	MFA	BA	BAM	BSO
1500	4.2541	4.5491	12.3797	20.5653	10.076	16.3819	6.2034	1.9576	16.6782	**1.4842**	2.7838
3000	3.5523	3.4353	6.0887	20.6706	9.1725	16.2884	4.3526	1.3048	14.9048	**0.9337**	1.2222
4500	3.4269	3.1636	3.7267	20.1996	9.5459	16.1408	4.1809	0.9933	14.4874	0.9123	**0.9062**
6000	3.0080	2.6434	2.6358	20.8610	8.9917	16.3976	2.2791	0.8984	12.4323	0.8000	**0.7533**
7500	2.9160	3.1409	1.9715	20.7600	7.8005	16.1958	2.2064	0.7025	11.4213	0.7422	**0.7014**

približan zaključak, i to na osnovu dobijenih najboljih, najgorih i prosečnih rezultata. S obzirom da su rezultati pokazali da je najmanja razlika između najboljeg i najgoreg rešenja dobijena primenom našeg predloženog algoritma, može se zaključiti da je u poređenju sa ostalim navedenim prirodom inspirisanim algoritmima naš predloženi BSO algoritam najstabilniji.

Drugi set eksperimenata se odnosio na testiranje algoritma sa različitim dimenzijama problema. Maksimalni broj evaluacija fitnes funkcije je postavljen na 6000 da bi algoritam bio uporediv sa ostalim pomenutim algoritmima. Algoritmi su testirani za sledeće dimenzije: 5, 10, 15, 20, 25, 30, 35 i 40. Dobijeni rezultati su prikazani u Tabelama 6.5, 6.6 i 6.7. Najbolji rezultati dobijeni u 100 izvršavanja su dati u Tabeli 6.5.

Tabela 6.5 Najbolji rezultati dobijeni u 100 izvršavanja sa različitim dimenzijama problema

D	GA	SGA	DE	ES	PSO	ACO	FA	MFA	BA	BAM	BSO
5	5.2471	9.9596	4.3568	12.3746	5.6082	10.1164	4.3585	4.3573	10.6909	4.3575	**0.3843**
10	1.5716	1.5498	1.3952	8.0656	2.1101	7.4746	1.3990	1.3966	2.3600	1.3953	**0.3690**
15	0.8299	0.9700	0.6204	7.7408	3.2257	9.8297	0.6172	0.6115	3.0757	0.6094	**0.3774**
20	0.8600	0.8426	0.4913	9.6276	2.3738	10.0836	0.4626	0.4552	2.3950	0.4679	**0.4541**
25	1.5243	1.3743	0.6265	12.3169	2.3740	11.5490	0.4908	0.4571	5.0173	0.4484	**0.3929**
30	1.7026	1.5147	1.1301	18.0090	3.6751	13.8615	0.6828	0.5160	7.2470	0.4671	**0.4411**
35	2.1602	1.5319	1.2849	16.8613	5.4765	16.9476	1.0829	**0.4709**	7.4484	0.4795	0.5979
40	2.4178	1.9406	3.9617	19.8244	5.5384	17.6142	1.5225	**0.4506**	8.6500	0.6028	0.6003

Naš predloženi metod je pronašao najbolja rešenja za dimenzije manje od 35, tj. za dimenzije 5, 10, 15, 20, 25 i 30. MFA algoritam se pokazao najboljim u dostizanju najboljih rešenja za dimenzije 35 i 40. Međutim, naš predloženi BSO algoritam je pokazao bolje performanse u odnosu na MFA kada je maksimalni broj evaluacija fitnes

Tabela 6.6 Najgori rezultati dobijeni u 100 izvršavanja sa različitim dimenzijama problema

D	GA	SGA	DE	ES	PSO	ACO	FA	MFA	BA	BAM	BSO
5	20.1888	22.6326	9.7959	62.1765	13.3267	12.6928	15.7395	12.4186	295.2557	10.2403	**0.3873**
10	6.3799	5.7899	12.4821	74.6665	23.2604	18.2565	6.7095	3.7858	58.7386	10.7242	**0.4372**
15	8.1499	9.9385	12.5250	50.3214	28.0228	10.9917	44.2763	3.8319	35.7454	10.1928	**0.6771**
20	9.4820	11.6024	18.8897	38.7234	34.7133	17.0266	28.9142	2.0279	33.7068	3.7420	**2.0108**
25	12.7971	16.0736	17.1415	33.4598	31.6741	12.2373	16.4518	3.7043	24.9265	**3.5192**	4.0914
30	22.1291	14.0512	29.6529	37.4566	35.6656	14.4647	15.9757	8.3364	30.0844	10.2851	**8.0677**
35	24.4790	15.6693	39.4435	46.6475	38.0578	18.7271	33.8871	**5.8830**	32.7374	8.8193	8.2009
40	19.2098	22.5022	45.4130	44.3624	35.5090	27.0641	36.6626	**7.7236**	33.2634	8.4273	8.2159

Tabela 6.7 Prosečni rezultati dobijeni u 100 izvršavanja sa različitim dimenzijama problema

D	GA	SGA	DE	ES	PSO	ACO	FA	MFA	BA	BAM	BSO
5	10.5709	10.8836	8.0557	31.8202	10.0765	11.4856	8.7499	9.1673	56.4830	9.0542	**0.3856**
10	2.3722	2.2813	3.1206	27.2252	7.2212	12.5333	2.1801	1.5740	19.4251	2.7075	**0.4304**
15	2.1136	1.8973	2.3737	22.0792	7.7362	10.2484	2.8217	0.8967	13.6018	1.2318	**0.4305**
20	2.9612	2.8621	3.0044	20.4717	9.9091	16.3303	3.7327	0.7004	13.6305	0.7609	**0.7003**
25	3.7244	3.7238	4.6029	22.7244	10.3315	11.5842	3.9039	0.9987	14.9017	0.7093	**0.6851**
30	5.3097	4.3798	11.4103	25.4016	12.7964	13.9422	4.9621	1.3568	16.6162	1.1067	**1.0530**
35	6.0765	5.4943	19.1074	27.2172	13.8799	18.3452	5.9955	1.6009	17.7033	1.4617	**1.4535**
40	7.6989	7.4237	28.7062	30.0177	15.1555	24.7642	7.8558	2.1978	19.9737	**1.8769**	2.0039

funkcije povećan na 10000. U tom slučaju, za dimenzije problema 35 i 40, primenom BSO algoritma su se dobile najbolje vrednosti 0.4418 i 0.4502, respektivno (a srednje vrednosti su tada bile 1.1495 i 1.4210, respektivno). Ovo je posledica osobine BSO algoritma da se veoma dobro ponaša u rešavanju višeciljnih i velikih optimizacionih problema, ali tada zahteva određeni broj iteracija koje nužno ne povećavaju vreme izračunavanja.

U većini slučajevima je naš algoritam imao najmanje loše rešenje u 100 izvršavanja (Tabela 6.6). Svi algoritmi osim MFA, BAM i našeg BSO imaju veoma loše najgore rezultate. Za dimenzije problema 5, 10 i 15, predloženi BSO algoritam je imao značajno manje najgore rešenje u poređenju sa BAM, dok u ostalim slučajevima razlika nije velika. U poređenju sa MFA, naš predloženi algoritam je imao značajno manju vrednost najgoreg rešenja kada je dimenzija problema bila 5, dok je za dimenzije 35 i 40 MFA algoritam dao bolje rezultate. U ostalim slučajevima, najlošija rešenja su bila slična.

U proseku, za 100 izvršavanja, naš predloženi algoritam je bolji od ostalih za sve

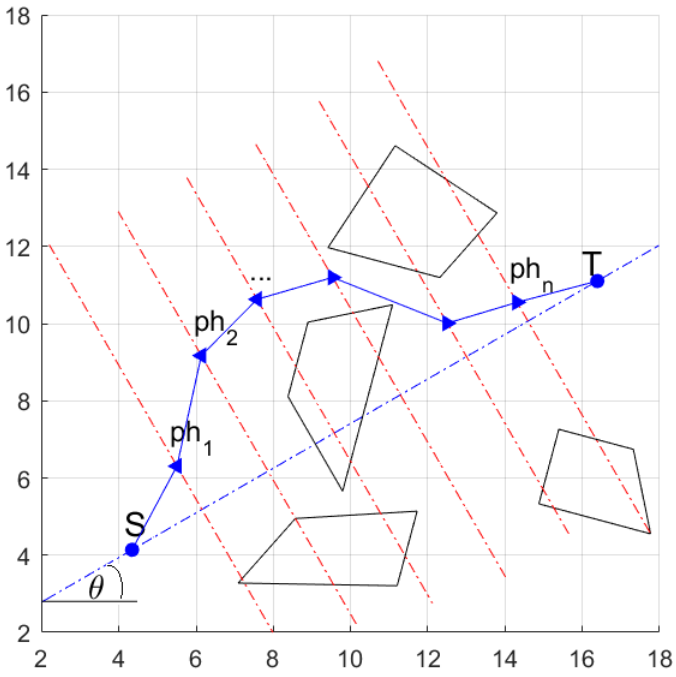
dimenzije problema osim u slučaju dimenzije 40 kada je BAM algoritam imao bolje performanse. Predloženi BSO algoritam je u proseku, za 100 izvršavanja i 7000 evaluacija fitnes funkcije, pronašao bolje rešenje u poređenju sa BAM, 1.6892 prema 1.8769. Ovaj rezultat dokazuje kvalitet predloženog brain storm optimizacionog algoritma za rešavanje problema planiranja putanje UCAV letelica, čak i za velike dimenzije. U ovim eksperimentima smo fiksirali broj evaluacija funkcije cilja kako bismo na fer način uporedili rezultate, ali je dobro poznato da je za velike dimenzije problema potrebno više iteracija.

Dodatno, izvršili smo kvalitativno poređenje našeg predloženog BSO algoritma sa metodom predloženim u [194], gde se za planiranje putanje UCAV letelica koriste veštačke neuralne mreže (artificial neural network, ANN) trenirane primenom imperijalističkog kompetitivnog algoritma (imperialist competitive algorithm, ICA). ICA-ANN algoritam je upoređivan sa algoritmom veštačke kolonije pčela testiranjem u dva različita test okruženja, i rezultati simulacije su pokazali da ICA-ANN ima bolje peformanse. Optimalna putanja je definisana korišćenjem iste fitnes funkcije kao u našem istraživanju. Test okruženja koja se koriste u [194] imaju 7 i 8 oblasti opasnosti, međutim, bez preciznih koordinata, tako da smo ih morali približno odrediti na osnovu slike. U oba slučaja, naš predloženi BSO algoritam je pronašao najbolju moguću pravu putanju, dok je ICA-ANN algoritam generisao rešenja koja imaju nepotrebne lukove i zaokrete prilikom izbegavanja zona opasnosti.

6.2 Planiranje putanje robota u neizvesnom okruženju primenom brain storm algoritma

6.2.1 Matematički model

U ovom istraživanju se razmatra kretanje robota u dvodimenzionalnom prostoru. Putanja robota se pretražuje od početne pozicije do ciljne pozicije, pri čemu je neophodno izbegavanje statičkih prepreka i neizvesnih izvora opasnosti. Problem planiranja robota se može definisati kao nalaženje optimalne rute bez kolizija, od početne pozicije (S) do ciljne pozicije (T), u skladu sa nekom metrikom. Optimalna putanja može biti najkraća ili najglatkija, putanja koja troši najmanje vremena ili energije robota, ili se može koristiti neki drugi kriterijum. U ovom istraživanju se razmatraju dužina



Slika 6.2 Ilustracija modela putanje

putanje i bezbednost prilikom pretraživanja optimalne rute robota.

Za modelovanje rešenja smo usvojili metod koji je korišćen u poglavlju 6.1, a koji je takođe korišćen u brojnim drugim radovima zato što je veoma prost i ima malu osetljivost na oblik prepreka. Opisani model je predstavljen Slikom 6.2.

6.2.2 Kriterijumi performansi

Cilj metoda za planiranje putanje je da se nađe optimalna ruta, pri čemu optimalnost može da ima različita značenja. Mi ćemo ovde korisiti dva kriterijuma: dužinu putanje i bezbednost.

Dužina putanje se može izračunati kao zbir rastojanja između dve susedne tačke na putanji. Izračunavanje dužine putanje, $L(PH)$, se vrši na potpuno isti način kao u poglavlju 6.1.

$$L(PH) = \sum_{i=0}^{n} d(ph_i, ph_{i+1}) \tag{6.10}$$

gde je $d(a, b)$ euklidsko rastojanje između a i b.

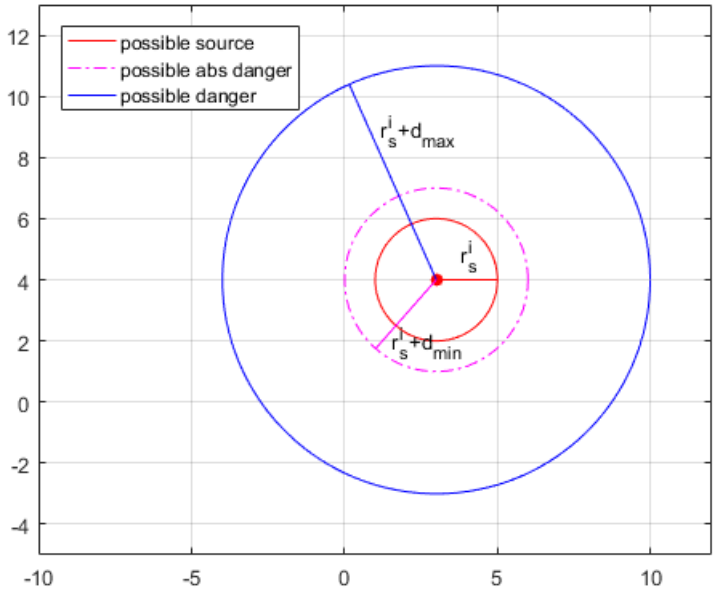
Dužina putanje je prilično jasna i jednostavna jer ne postoje nepoznati ili ne-

izvesni faktori. Međutim, stepen bezbednosti može zavisiti od nekoliko neizvesnih parametara. Razvili smo probabilistički model za izračunavanje stepena bezbednosti u neizvesnom okruženju koji može biti precizniji od fazi modela u [148]. Statičke prepreke su fiksne i putanja jedino mora da ide oko njih, zato što robot ne može da ide kroz njih. S druge strane, postoje zone opasnosti koje nisu fizičke prepreke, ali koje mogu da oštete ili pokvare robota. Na primer, zone opasnosti mogu biti zone visoke radijacije, toplote, hladnoće, izloženosti neprijateljskoj vatri i slično. Zone opasnosti imaju svoje izvore, i opasnost je veća u blizini izvora, a opada sa porastom rastojanja od izvora. Različiti izvori mogu imati različite funkcije njihovog stepena opasnosti. U ovom istraživanju, mi koristimo funkciju za stepen opasnosti koja je predložena u [155], i koristi se u [148]. U zavisnosti od najkraćeg rastojanja $d(DS_i)$ između izvora opasnosti i i putanje, stepen opasnosti dan_i se može definisati na sledeći način:

$$dan_i = \begin{cases} 1, ako \quad d(DS_i) \leq d^i_{min} \\ \frac{d^i_{max} - d(DS_i)}{d^i_{max} - d^i_{min}}, ako \quad d^i_{min} < d(DS_i) < d^i_{max} \\ 0, ako \quad d(DS_i) \geq d^i_{max} \end{cases} \tag{6.11}$$

U prethodnoj funkciji, d^i_{min} i d^i_{max} su definisani za svaki izvor opasnosti i oni predstavljaju granice zona apsolutne opasnosti i zona slobodnih od opasnosti, respektivno. Ako je najkraće rastojanje između putanje i izvora opasnosti veće od d^i_{max}, onda je robot apsolutno bezbedan. Ako je to rastojanje manje od d^i_{min}, onda je robot u apsolutnoj opasnosti. U zoni između, stepen opasnosti se računa na osnovu rastojanja i razlike između d^i_{min} i d^i_{max}.

Tačna pozicija izvora opasnosti nije uvek poznata, ali zona gde se izvor nalazi obično može biti određena. U ovom istraživanju se analizira model za planiranje putanje gde su pozicije izvora opasnosti unutar kružnih oblasti. U [148] je predložen fazi model za izračunavanje stepena opasnosti u neizvesnom okruženju. U ovakvim problemima planiranja putanje, verovatnoća da je tačka na putanji u opasnosti može biti direktno izračunata i tačnost ovakvog postupka je veća nego u slučaju fazi modela. Ako je poznato da je izvor opasnosti negde unutar kruga poluprečnika r^i_s, i ako je stepen opasnosti izračunat primenom jednačine 6.11, uočava se da putanja može biti u opasnosti ako se nalazi unutar velikog kruga poluprečnika $r^i_s + d^i_{max}$, a centar je isti kao centar kruga mogućeg položaja izvora (Slika 6.3).



Slika 6.3 Zona opasnosti oko neizvesnog izvora opasnosti

Zona opasnosti predstavlja geometrijsko mesto svih krugova poluprečnika d^i_{max} i centrima u krugu poluprečnika r^i_s i centrom u tački (x_c, y_c). Na Slici 6.3 crveni krug je oblast gde može biti izvor opasnosti. Izvan plavog kruga na Slici 6.3 putanja je potpuno bezbedna, bez obzira na poziciju izvora opasnosti. Stepen opasnosti za tačku unutar plavog kruga zavisi od rastojanja i pozicije izvora opasnosti. On se definiše kao integral funkcije stepena opasnosti preko svih mogućih pozicija izvora (za sve moguće vrednosti x i y koordinata unutar crvenog kruga). Dakle, za izračunavanje stepena opasnosti neophodno je rešavanje Stieltjes-ovog integrala, zato što verovatnoće pozicije izvora opasnosti unutar date oblasti imaju uniformnu raspodelu. U slučaju drugačije raspodele pozicije izvora opasnosti unutar crvenog kruga, umesto Stieltjes-ovog integrala bi se koristio Lebesque-ov integral sa tom funkcijom raspodele kao merom. Stepen opasnosti u tački (x_0, y_0) se dobija rešavanjem sledećeg integrala:

$$
\begin{aligned}
&dan^i(x_0, y_0) \\
&= I_{\sqrt{(x-x_0)^2+(y-y_0)^2} \le d^i_{min}} \int_{a_x}^{b_x} \int_{a_y}^{b_y} dydx \\
&\quad + I_{d^i_{min} \le \sqrt{(x-x_0)^2+(y-y_0)^2} \le d^i_{max}} \int_{a_x}^{b_x} \int_{a_y}^{b_y} \frac{d^i_{max} - \sqrt{(x-x_0)^2+(y-y_0)^2}}{d^i_{max} - d^i_{min}} dydx
\end{aligned}
\tag{6.12}
$$

Dva integrala su neophodna zato što model za izračunavanje stepena opasnosti ima tri slučaja, ali poslednji je jednak nuli, tako da on ne doprinosi sumi. Uvođenjem indikatora I samo odgovarajući integral neće biti postavljen na nulu, tako da će jedan integral uvek biti nula. Granice integrala su označene kao: a_x, b_x, a_y i b_y, gde se a_x i b_x koriste kao granice za x koordinatu, dok su a_y i b_y zavisne od x, i koriste se kao granice za y koordinatu. Granice za x uzimaju sledeće vrednosti:

$$\begin{aligned} a_x &= x_c - r_s^i \\ b_x &= x_c + r_s^i \end{aligned} \tag{6.13}$$

gde je x_c x koordinata centra oblasti gde se može nalaziti izvor. S obzirom da je oblast integracije krug čiji centar nije u tački $(0, 0)$, već u (x_c, y_c), on se definiše na sledeći način:

$$(x - x_c)^2 + (y - y_c)^2 = (r_s^i)^2 \tag{6.14}$$

Izražavanjem y iz prethodne jednačine dobijaju se granice a_y i b_y:

$$\begin{aligned} a_y &= \sqrt{(r_s^i)^2 - (x - x_c)^2} + y_c \\ b_y &= -\sqrt{(r_s^i)^2 - (x - x_c)^2} - y_c \end{aligned} \tag{6.15}$$

Stepen opasnosti je određen primenom našeg matematičkog modela, korišćenjem jednačine 6.12. S obzirom da je probabilistički model upotrebljavan za izračunavanje stepena opasnosti u slučaju jednog izvora opasnosti, njihovo kombinovanje je jednostavno primenom teorije verovatnoće. Ako je jedna tačka pod uticajem dve ili više preklapajućih zona opasnosti, stepen opasnosti se izračunava korišćenjem činjenice da je verovatnoća da tačka nije pod uticajem niti jedne zone opasnosti jednaka proizvodu verovatnoća da tačka nije pod uticajem svake od zone pojedinačno, pod uslovom da su one nezavisne.

$$D(x_0, y_0) = 1 - (1 - dan^i(x_0, y_0))(1 - dan^j(x_0, y_0)) \tag{6.16}$$

Na analogan način se izračunava i u slučaju više zona opasnosti koje se presecaju.

U zavisnosti od svakog određenog izvora opasnosti, stepen rizika mora biti izračunat na različit način. U nekim okruženjima se rizik povećava sa vremenom provedenim u

zoni opasnosti, dok se u ostalim razmatra dužina putanje koja ide kroz rizičnu oblast. Na primer, u slučaju radijacije, vreme provedeno u oblasti zračenja je odlučujući faktor, zajedno sa snagom radijacije. U našem modelu snaga radijacije se predstavlja verovatnoćom opasnosti, i ukupan rizik se može izračunati integraljenjem duž putanje u odnosu na brzinu. Ako se robot kreće kroz prašinu ili neki drugi izvor zagađenja, onda je važna samo dužina putanje, ali ne i brzina.

U ovom istraživanju smo kao estimator opasnosti koristili tačku koja je najbliža izvoru opasnosti, da bismo mogli vršiti upoređivanje sa modelom u [148]. Ovaj model je pogodan za situacije slične pretnji "jednog projektila", kada dužina putanje kroz zonu opasnosti ne utiče na nivo pretnje; projektil je ispaljen samo jednom i on će pogoditi ili promašiti sa datom verovatnoćom (koja je zavisna od blizine). Za razliku od modela u [148], naš predloženi metod može biti lako prilagođen ostalim opisanim situacijama.

6.2.3 Predloženi algoritam

U ovom istraživanju se predlaže metod za planiranje putanje robota u neizvesnom okruženju. Osim statičkih prepreka, u problem planiranja putanje uključeni su i izvori opasnosti sa neizvesnim pozicijama. Napravili smo metod za planiranje putanje koji minimizuje dužinu putanje i maksimizuje bezbednost. Ova dva cilja su u većini slučajeva kontradiktorni, tj. ako je putanja minimalna, ona ide pravo kroz zonu opasnosti, tako da ako se želi povećanje bezbednosti (ili smanjenje stepena opasnosti) mora se takođe povećati i dužina putanje. Za višeciljne optimizacione probleme su u prošlosti predlagani različiti pristupi. U ovom istraživanju naša ciljna funkcija uključuje parametar w koji kontroliše uticaj svakog od kriterijuma. S obzirom da su oni međusobno suprotni, kontrolni parametar daje veću važnost nekome od njih.

Osim ova dva cilja, rešenje takođe mora biti i izvodljivo. U cilju implementacije ovog ograničenja uveli smo kaznu (penalty) u ciljnu funkciju, koja ima veću vrednost od vrednosti dužine putanje i stepena opasnosti. Da bi se smanjio broj iteracija potrebnih za nalaženje izvodljivog rešenja i da bi se sprečilo zaglavljivanje u lokalnom optimumu koje je dobijeno od strane neizvodljivih rešenja, implementirali smo više eksploracije ako je neizvodljivo rešenje globalno najbolje u nekoliko generacija. Obično metode za planiranje putanje robota uvode neke mehanizme za generisanje izvodlji-

vih rešenja na osnovu neizvodljivih rešenja. U [148] se predlaže ograničena petlja za ažuriranje rešenja. Novo rešenje iz neizvodljivog rešenja se generiše korišćenjem metoda zasnovanog na slučajnom semplovanju i uniformnoj mutaciji. Ovaj mehanizam može da smanji broj iteracija, ali on povećava vreme izračunavanja koje je važan resurs optimizacionih algoritama. S druge strane, naš predloženi brain storm optimizacioni pristup samo povećava eksploraciju, što ne utiče na vreme izračunavanja iteracija, niti na složenost algoritma.

Sledeće parametre smo odredili na empirijski način. Ako je rešenje neizvodljivo u više od 5 generacija, verovatnoća p_{5a} koja kontroliše zamenu centra klastera novim rešenjem, se povećava sa 0.2 na 0.5. Dodatno, ako se izvodljivo rešenje ne nađe u više od 10 generacija, p_{5a} se postavlja na 0.9, što znači da u 90% slučajeva najbolje rešenje u jednom klasteru će biti zamenjeno slučajnim rešenjem. Ovo značajno naglašava eksploraciju i smanjuje broj generacija potrebnih za nalaženje izvodljivog rešenja. Bez ovog dela, u nekim slučajevima BSO nije mogao da nađe izvodljivo rešenje u 500 iteracija, dok uz ovaj mehanizam nije potrebno više od 100 iteracija da bi se našlo izvodljivo rešenje. U većini slučajeva, izvodljiva rešenja su nađena u prvih 50 iteracija.

Ciljna funkcija sadrži normalizovanu dužinu putanje definisanu jednačinom 6.10, stepen opasnosti iz jednačine 6.12 i kaznu za neizvodljiva rešenja. Stepen opasnosti je u opsegu [0,1], tako da dužina putanje mora da bude normalizovana. S obzirom da najduža putanja ne može biti izračunata, koristili smo sledeću aproksimaciju. Usvojeno je da je najduža putanja dva puta duža od najkraće putanje. Sa tom pretpostavkom, dužina putanje se normalizuje na sledeći način:

$$L(PH)_{normalizovano} = \frac{L(PH) - d(S,T)}{d(S,T)} \tag{6.17}$$

gde je $d(S,T)$ euklidsko rastojanje između početne tačke S i ciljne tačke T. Kombinovanjem prethodno pomenutih kriterijuma, sledeća ciljna funkcija je korišćena u brain storm optimizacionom algoritmu:

$$F(PH) = wL(PH) + (1-w)D(PH) + kazna \tag{6.18}$$

Putanja robota je definisana sa $n+2$ tačaka, uključujući početnu i cijnu poziciju. Dimenzija brain storm algoritma je n zato što je samo potreban ofset od x-ose da

bi se odredila tačka putanje ph_i. Prve koordinate od ph_i se izračunavaju jednom na početku, a susedne tačke su ekvidistantne.

6.2.4 Rezultati simulacije

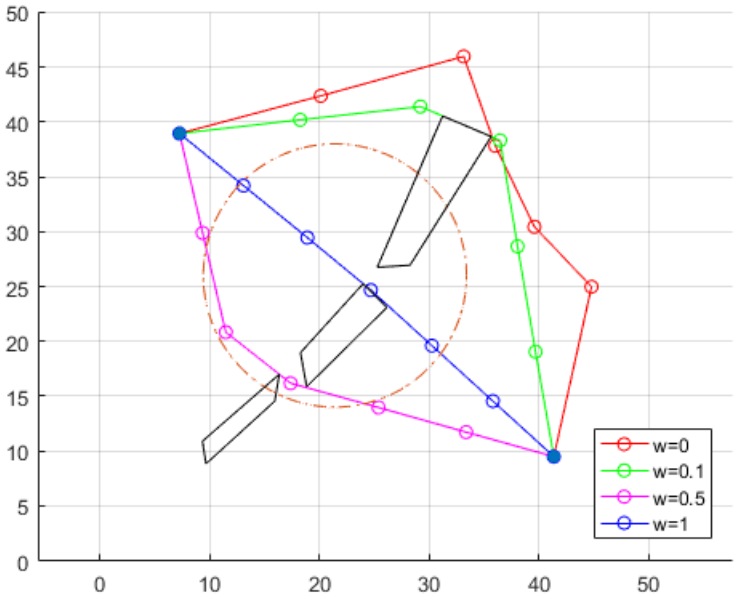
Predloženi metod za planiranje putanje robota u neizvesnom okruženju je implementiran u Matlabu, verzija R2016b. Sve simulacije su izvršene na računarskoj platformi Intel Core i7-3770K CPU 4GHz, 8GB RAM, sa operativnim sistemom Windows 10 Professional OS.

U literaturi se mogu naći brojni test primeri za planiranje putanje. U ovom istraživanju smo koristili primere predložene u [148], gde su kreirana četiri različita test okruženja za testiranje algoritama za planiranje putanje robota. Test primeri sadrže različiti broj statičkih prepreka, kao i izvora opasnosti. Za sve izvore opasnosti, vrednosti d^i_{min} i d^i_{max} su postavljene na 1 i 10, respektivno. Veličina oblasti pretraživanja je 50×50. Za sve test primere naš predloženi metod je izvršavan 30 puta. Detaljne informacije o svakom test problemu su date u nastavku ovog poglavlja. Parametri brain storm algoritma su postavljeni empirijskim putem. Broj agenata je 100, a broj klastera je 5. Verovatnoća p_{5a} je inicijalno postavljena na vrednost 0.2, ali se može menjati, kao što je opisano u poglavlju 6.2.3. Verovatnoće p_{6b}, p_{6bii} i p_{6c} su postavljene na vrednosti 0.8, 0.4 i 0.5, respektivno. Kazna za neizvodljiva rešenja je 20. Maksimalni broj iteracija je bio 500.

Naš predloženi algoritam je upoređivan sa [148], gde je korišćen Pareto front za rešavanje višeciljnih optimizacionih problema. Za stepen opasnosti je u [148] korišćen fazi model, koji nije predstavljen jednom vrednošću već intervalom. U ovom istraživanju smo koristili naš predloženi matematički model, tako da je stepen opasnosti jedistveno određen za svaku putanju, što preciznije opisuje istu situaciju. U [148] se pored dužine putanje i stepena opasnosti koristi i intervalski poboljšana hipervolume metrika, napravljena da radi sa nepreciznim optimizacionim problemima. S obzirom da mi zbog predloženog probabilističkog modela za stepen opasnosti nemamo nepreciznu optimizaciju, ova metrika nije uključena.

Test primer 1

Prvi test primer sadrži tri statičke prepreke i jedan neizvesni izvor opasnosti. Početna pozicija je $(7.2399, 38.9460)$, dok su koordinate cilja $(41.3780, 9.4897)$. Položaj



Slika 6.4 Optimalne putanje za prvi test problem, dobijene primenom BSO sa različitim vrednostima parametra w

izvora opasnosti je u krugu poluprečnika 2 i centra (21.4110, 26.0000). Koordinate prepreka su sledeće:

$$ob1 = \begin{bmatrix} 23.987 & 25.250 \\ 18.271 & 18.964 \\ 18.834 & 15.869 \\ 26.161 & 23.092 \end{bmatrix} \quad ob2 = \begin{bmatrix} 25.275 & 26.750 \\ 31.233 & 40.54 \\ 35.662 & 38.664 \\ 28.254 & 26.938 \end{bmatrix} \quad ob3 = \begin{bmatrix} 16.338 & 16.994 \\ 9.3333 & 10.897 \\ 9.6554 & 8.833 \\ 15.936 & 14.555 \end{bmatrix}$$

Broj segmenata putanje je u svim test problemima isti kao u [148]. U prvom test primeru je jednak 6, što znači da je dimenzija problema 5. Na Slici 6.4 su prikazai rezultati za različite vrednosti parametra w.

Kao što se može videti sa Slike 6.4, ako parametar ciljne funkcije w ima vrednost 1, što znači da se razmatra samo dužina putanje, naš predloženi metod nalazi pravu liniju (plava putanja na Slici 6.4) između početne i ciljne tačke, što je i najkraća putanja. S druge strane, ako w ima vrednost 0, tj. ako se treba minimizovati samo stepen opasnosti, može se uočiti da je zona opasnosti u potpunosti izbegnuta (crvena putanja

na Slici 6.4), ali je dužina putanje nepotrebno dugačka zbog činjenice da uopšte nije ni razmatrana. Ako w ima vrednost 0.1, što znači da se minimizacija putanje pokušava tek ako je zona opasnosti skoro u potpunosti izbegnuta, predloženi algoritam nalazi zelenu putanju na Slici 6.4. Ova putanja je izbegla zonu opasnosti i smanjila dužinu putanje u odnosu na prethodni slučaj. Kada se vrednost parametra w povećala na 0.5, dužina putanje i stepen opasnosti su jednako razmatrani, a putanja koja je dobijena primenom našeg predloženog metoda je prikazana ljubičastom bojom na Slici 6.4. Ova putanja je kraća u odnosu na prethodnu, ali je i dalje izbegnuta zona opasnosti. Ovaj test problem ima dobru mogućnost za prikazivanje kvaliteta predloženog rešenja. U zavisnosti od potreba, dužina ili bezbednost putanje se mogu razmatrati kao manje ili više važni ciljevi, i predloženi metod će naći najbolje rešenje u datoj situaciji.

Rezultati objavljeni u [148] za slučaj prvog test problema imaju određenu nekonzistentnost. U radu se kaže da je linija između početne i ciljne pozicije podeljena na jednake segmente, međutim, rešenje na slici koje ima tačke u uglovima prepreka nije moguće. Ovo rešenje je prijavljeno kao rešenje sa minimalnim stepenom opasnosti i ekvivalentno je našem rešenju kada je w jednako 0.1.

U Tabeli 6.8 su prikazane dužine putanje i stepeni opasnisti za 10 različitih vrednosti parametra w, formirajući Pareto front. Sledeći uobičajenu praksu kod algoritama inteligencije rojeva, date su srednje vrednosti, standardne devijacije i najbolja rešenja.

Naša najkraća putanja ($w = 1$) je ista kao u [148], prava linija od početne do ciljne tačke. S druge strane, kada je w jednako 0, dužina putanja se uopšte ne razmatra. U tom slučaju je lako da se rizik svede na nulu jer se putanja svodi na proizvoljno kretanje kroz oblast pretraživanja uz izbegavanje samo zona opasnosti, te se tada očekuje velika standardna devijacija. Kada se vrednost w poveća na 0.1, putanja se stabilizuje pri čemu se rizik zadržava na nuli. Iz rezultata prikazanih u Tabeli 6.8 može se zaključiti da naš predloženi metod kreira dobar Pareto front i da je algoritam robustan sa malom standardnom devijacijom.

U [148] su dobijene putanje dužine 58.5849 i 45.1079 sa stepenima opasnosti 0.0 i 0.7356 do 1.0, respektivno. Naša najbolja putanja sa rizikom jednakim nuli je ona koja se nalazi iznad svih prepreka (zelena putanja na Slici 6.4). U drugom slučaju, naše rešenje je takođe prava linija (plava putanja na Slici 6.4), a vrednost rizika je izračunata u skladu sa našim probabilističkim modelom i iznosi 0.926.

Tabela 6.8 Dužine putanja i stepeni opasnosti za prvi test primer kada se koriste različite vrednosti parametra w

	L(PH)			D(PH)		
w	mean	std	best	mean	std	best
0.0	79.868	13.566	68.174	0.000	0.000	0.000
0.1	59.241	0.000	59.241	0.000	0.000	0.000
0.2	58.465	2.453	51.483	0.003	0.009	0.000
0.3	52.901	3.342	51.303	0.023	0.012	0.000
0.4	51.177	0.004	51.170	0.030	0.000	0.030
0.5	51.044	0.002	51.042	0.033	0.000	0.032
0.6	50.902	0.001	50.900	0.036	0.000	0.036
0.7	50.748	0.000	50.748	0.043	0.000	0.043
0.8	50.358	0.000	50.358	0.067	0.000	0.067
0.9	45.482	0.000	45.481	0.743	0.000	0.743
1.0	45.108	0.000	45.108	0.926	0.000	0.926

Test primer 2

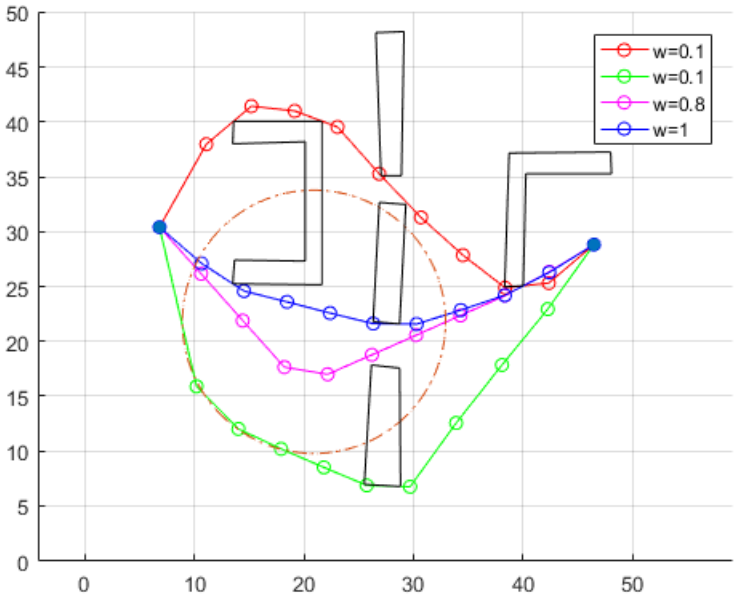
Drugi test primer sadrži tri konveksne statičke prepreke i dve nekonveksne statičke prepreke, pri čemu je jedna U-oblika, a druga je V-oblika. Test primer sadrži takođe jedan neizvesni izvor opasnosti. Početne i ciljne pozicije su (6.8374, 30.409) i (46.451, 28.814), respektivno. Pozicija izvora opasnosti je u krugu poluprečnika 2 sa centrom u (20.928, 21.779). Koordinate prepreka su sledeće:

$$ob1 = \begin{bmatrix} 13.601 & 21.652 & 21.652 & 13.520 & 13.681 & 20.122 & 20.122 & 13.52 \\ 40.071 & 40.071 & 25.156 & 25.250 & 27.407 & 27.313 & 38.195 & 38.00 \end{bmatrix}$$

$$ob2 = \begin{bmatrix} 47.981 & 38.721 & 38.319 & 40.010 & 40.251 & 48.061 \\ 37.257 & 37.163 & 24.968 & 25.062 & 35.287 & 35.287 \end{bmatrix}$$

$$ob3 = \begin{bmatrix} 26.564 & 48.139 \\ 27.047 & 35.099 \\ 28.899 & 35.099 \\ 29.140 & 48.233 \end{bmatrix} \quad ob4 = \begin{bmatrix} 26.886 & 32.660 \\ 26.322 & 21.779 \\ 28.738 & 21.591 \\ 29.301 & 32.473 \end{bmatrix} \quad ob5 = \begin{bmatrix} 26.161 & 17.839 \\ 25.517 & 6.9568 \\ 28.818 & 6.7692 \\ 28.738 & 17.557 \end{bmatrix}$$

U drugom primeru, putanja je podeljena u 10 segmenata, što znači da je dimenzija



Slika 6.5 Optimalne putanje za drugi test problem, dobijene primenom BSO sa različitim vrednostima parametra w

problema 9. Slično kao u prvom primeru, na Slici 6.5 su prikazane putanje dobijene našim predloženim metodom, za različite vrednosti parametra w. Plava putanja predstavlja najkraću putanju pronađenu našim predloženim metodom kada je stepen opasnosti bio zanemaren. U ovom primeru, prava linija nije izvodljivo rešenje, ali se može uočiti da dobijena putanja prati ivice prepreka, i između njih putanja je prava linija. Zelena i crvena putanja su za slučaj $w = 0.1$. Zelena putanja prati ivice zone opasnosti izbegavajući istu, kao i ivicu prepreke ispod nje. S druge strane, drugo rešenje za $w = 0.1$ je pronađeno i ono izbegava zonu opasnosti odozgo. Optimalna putanja dobijena za $w = 0.8$ ide kroz zonu opasnosti uz malo skretanje u cilju izbegavanja najopasnije oblasti.

U Tabeli 6.9 su prikazane dužine putanja i stepeni opasnosti za drugi test primer, za $w \in \{0, 0.1, 0.2, \ldots, 1\}$.

S obzirom da naš predloženi metod nalazi putanju minimalne dužine za $w = 1$, sa standardnom devijacijom 0, može se zaključiti da je predloženi metod sposoban za nalaženje najkraće putanje na robustan način. U ovom primeru, standardna devijacija je veoma velika i za stepen rizika i za dužinu putanje, kada w ima vrednost 0.9, 0.8

Tabela 6.9 Dužine putanja i stepeni opasnosti za drugi test primer kada se koriste različite vrednosti parametra w

	L(PH)			D(PH)		
w	mean	std	best	mean	std	best
0.0	85.1425	15.942	72.226	0.000	0.000	0.000
0.1	63.617	3.787	52.838	0.000	0.000	0.000
0.2	64.914	3.333	52.804	0.000	0.000	0.000
0.3	64.634	3.321	52.509	0.001	0.000	0.000
0.4	63.404	3.767	52.732	0.001	0.001	0.000
0.5	63.245	3.710	52.745	0.004	0.002	0.000
0.6	64.195	0.496	64.000	0.010	0.004	0.000
0.7	58.245	7.012	50.073	0.191	0.211	0.026
0.8	54.142	7.839	47.941	0.386	0.282	0.056
0.9	52.162	10.908	43.666	0.529	0.426	0.000
1.0	43.378	0.001	43.378	0.921	0.003	0.917

i 0.7. Razlog ovome je priroda ovog test primera, jer ovde najkraća putanja prolazi kroz oblast gde se nalazi centar izvora opasnosti, te je stoga vrednost stepena rizika visoka i iznosi 0.921. S druge strane, izvodljive putanje koje nisu tako blizu centra opasnosti su samo one koje kompletno izbegavaju oblast rizika, ali su znatno duže. Stoga, dve klase nedominiranih rešenja imaju veoma slične vrednosti ciljne funkcije, i u 30 izvršavanja se pojavljuje njihova kombinacija. U [148] je nađena putanja dužine 51.4852 kao najbezbednija, sa stepenom opasnosti 0.0. Naš predloženi algoritam je našao putanju dužine 52.509 i stepena opasnosti 0 (slična crvenoj putanji na Slici 6.5). Ponovo, rešenje dobijeno u [148] ide direktno duž ivica prepreka što nije moguće ako je putanja podeljena u jednake segmente, kao što je neophodno. Najkraća putanja u [148] ima dužinu 43.3782 i stepen opasnosti od 0.7329 do 1.0. Naš algoritam je našao istu optimalnu putanju kada je stepen rizika ignorisan (plava putanja na Slici 6.5). U ovom slučaju je stepen rizika je bio 0.921.

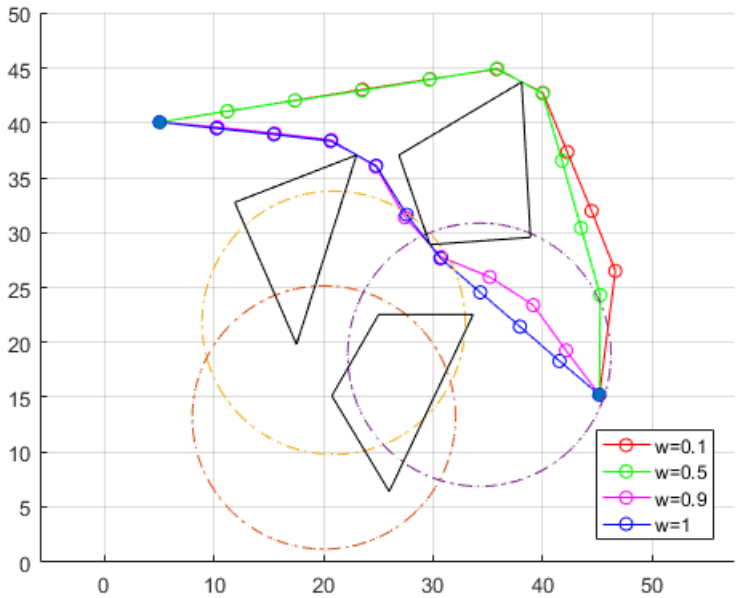
Test primer 3

Treći test primer sadrži tri statičke prepreke i tri neizvesna izvora opasnosti. Koordinate startne pozicije su (5.066, 40.071), dok su koordinate cilja (45.163, 15.212). Poluprečnik mogućih pozicija za sva tri izvora opasnosti je 2. Centar prvog izvora

opasnosti je (20.042, 13.148), centar drugog izvora je (20.928, 21.779), dok je centar trećeg izvora opasnosti u krugu sa centrom (34.213, 18.871). Pozicije prepreka su sledeće:

$$ob1 = \begin{bmatrix} 23.021 & 37.069 \\ 11.910 & 32.754 \\ 17.546 & 19.809 \end{bmatrix} \quad ob2 = \begin{bmatrix} 26.886 & 37.069 \\ 38.077 & 43.730 \\ 38.882 & 29.565 \\ 29.704 & 28.908 \end{bmatrix} \quad ob3 = \begin{bmatrix} 26.000 & 6.394 \\ 20.767 & 15.118 \\ 25.034 & 22.529 \\ 33.649 & 22.529 \end{bmatrix}$$

Na Slici 6.6 su prikazane putanje dobijene primenom predloženog metoda, sa različitim vrednostima parametra w. Broj segmenata je ponovo bio 10, a dimen-



Slika 6.6 Optimalne putanje za treći test problem, dobijene primenom BSO sa različitim vrednostima parametra w

zija problema 9. Ovaj primer ima samo tri prepreke i najkraća putanja se nalazi bez problema. Kada je $w = 0.9$, putanja je slična najkraćoj, ali je malo duža jer se želi izbeći izvor opasnosti. Ciljna tačka je unutar zone opasnosti, tako da mali deo putanje mora da bude unutar zone. U slučaju kada je $w = 0.1$ i $w = 0.5$, putanje su veoma slične. Jedina razlika je u poslednja četiri segmenta. Kada je bezbednost smatrana

važnijim ciljem, poslednji segment putanje teži da što kasnije uđe u zonu opasnosti.

Tabela 6.10 sadrži rezultate dobijene primenom predloženog algoritma za treći test problem.

Tabela 6.10 Dužine putanja i stepeni opasnosti za treći test primer kada se koriste različite vrednosti parametra w

	L(PH)			D(PH)		
w	mean	std	best	mean	std	best
0.0	86.0588	19.091	72.516	0.001	0.000	0.001
0.1	66.116	4.298	64.756	0.003	0.000	0.003
0.2	64.461	0.005	64.458	0.004	0.000	0.004
0.3	64.385	0.443	64.289	0.006	0.000	0.005
0.4	64.190	0.002	64.188	0.008	0.000	0.006
0.5	64.102	0.009	64.091	0.008	0.000	0.008
0.6	65.681	3.451	64.035	0.027	0.038	0.009
0.7	63.019	6.035	52.675	0.064	0.071	0.011
0.8	61.942	5.052	52.188	0.080	0.128	0.014
0.9	56.983	6.942	50.356	0.296	0.300	0.020
1.0	55.240	7.036	49.830	0.486	0.330	0.037

U [148] najkraća putanja ima dužinu 49.8304, i ista putanja je nađena primenom našeg predloženog metoda. Stepen opasnosti dobijen u [148] je bio u opsegu od 0.8359 do 1, dok je našim metodom izračunat stepen rizika od 0.486 (plava putanja na Slici 6.6). Velika razlika je zbog činjenice da u [148] drugi izvor opasnosti ima koordinate iste kao u našem istraživanju, (20.928, 21.779). Međutim, sa Slike 11 u [148] izgleda kao da su koordinate blizu (25,35) bile korišćene za drugi izvor opasnosti. U tom slučaju, naš algoritam bi izračunao rizik od 0.983. S druge strane, kada stepen rizika treba da bude minimizovan, u [148] je pronađena putanja dužine 64.9518 i stepena opasnosti od 0 do 0.0406, kao i putanja istog stepena opasnosti i dužine 65.985. Naš predloženi metod je našao putanju stepena rizika 0.008 i dužine 64.091 (zelena putanja na Slici 6.6), što je bolje od [148] i u pogledu dužine i u pogledu bezbednosti. Dodatno, nađena je i putanja stepena opasnosti 0.003 i dužine 64.756 (crvena putanja na Slici 6.6).

Test primer 4

Poslednji test primer ima trinaest statičkih prepreka i dva izvora opasnosti. U [148] nisu date precizne koordinate za ovaj test primer, te smo ih aproksimirali na osnovu slike. Početnu poziciju smo postavili u (7, 21), dok je ciljna pozicija imala koordinate (44, 29). Pozicije izvora opasnosti su u krugu poluprečnika 2, dok su centri u (23, 25) i (35, 33), respektivno. Koordinate prepreke su sledeće:

$$ob1 = \begin{bmatrix} 7.50 & 13.00 \\ 12.50 & 12.00 \\ 13.75 & 15.10 \\ 10.00 & 16.80 \end{bmatrix} \quad ob2 = \begin{bmatrix} 10.00 & 22.50 \\ 10.00 & 27.00 \\ 14.70 & 26.50 \\ 14.70 & 22.50 \end{bmatrix} \quad ob3 = \begin{bmatrix} 10.90 & 32.00 \\ 11.50 & 35.00 \\ 17.00 & 37.00 \\ 17.00 & 31.00 \end{bmatrix}$$

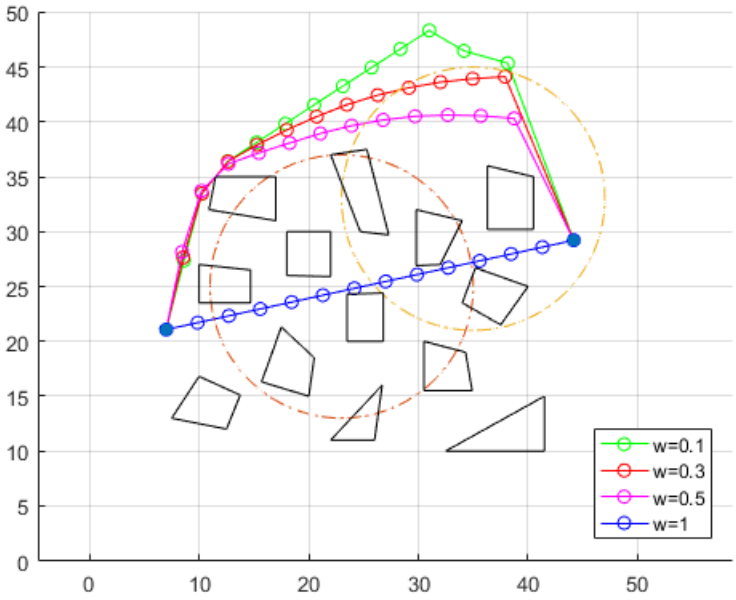
$$ob4 = \begin{bmatrix} 18.00 & 26.00 \\ 18.00 & 30.00 \\ 22.00 & 30.00 \\ 22.00 & 25.90 \end{bmatrix} \quad ob5 = \begin{bmatrix} 22.00 & 37.00 \\ 25.30 & 37.50 \\ 27.30 & 29.70 \\ 24.70 & 30.00 \end{bmatrix} \quad ob6 = \begin{bmatrix} 23.50 & 20.00 \\ 23.50 & 25.00 \\ 25.80 & 25.00 \\ 25.80 & 20.00 \end{bmatrix}$$

$$ob7 = \begin{bmatrix} 22.00 & 11.00 \\ 26.00 & 11.00 \\ 26.70 & 16.00 \end{bmatrix} \quad ob8 = \begin{bmatrix} 32.50 & 10.00 \\ 41.50 & 10.00 \\ 41.50 & 15.00 \end{bmatrix} \quad ob9 = \begin{bmatrix} 30.50 & 15.50 \\ 30.50 & 20.00 \\ 34.30 & 19.00 \\ 34.90 & 15.50 \end{bmatrix}$$

$$ob10 = \begin{bmatrix} 20.00 & 15.00 \\ 20.50 & 18.50 \\ 17.50 & 21.30 \\ 15.70 & 16.30 \end{bmatrix} \quad ob11 = \begin{bmatrix} 34.00 & 23.50 \\ 37.50 & 21.50 \\ 40.00 & 25.00 \\ 35.20 & 26.70 \end{bmatrix}$$

$$ob12 = \begin{bmatrix} 29.80 & 26.90 \\ 29.80 & 32.00 \\ 34.00 & 31.00 \\ 32.00 & 27.00 \end{bmatrix} \quad ob13 = \begin{bmatrix} 36.30 & 30.20 \\ 36.30 & 36.00 \\ 40.50 & 35.00 \\ 40.50 & 30.20 \end{bmatrix}$$

Putanja je podeljena u 13 delova tako da je dimenzija problema jednaka 12. Ovaj



Slika 6.7 Optimalne putanje za četvrti test problem, dobijene primenom BSO sa različitim vrednostima parametra w

poslednji primer je i najsloženiji. On sadrži 13 statičkih prepreka raspoređenih u centru polja. Jedan izvor opasnosti je takođe u centru, a drugi je izvor zone opasnosti u kojoj se nalazi ciljna pozicija. Najkraća putanja koja prolazi kroz zonu opasnosti je nađena i na Slici 6.7 je obeležena plavom linijom. Dodavanjem malog uticaja dužine putanje, da bi se sprečilo široko obilaženje kroz prostor pretraživanja, dobijena je putanja obeležena zelenom linijom. Ako je uticaj oba kriterijuma jednak, tj. ako je $w = 0.5$, dobijena putanja ide iznad prepreka, kako bi se smanjila dužina putanje, ali takođe pravi širok zaokret u cilju smanjivanja stepena opasnosti. Crvenom bojom je obeležena putanja za slučaj $w = 0.3$. Ovaj primer je dobra ilustracija uticaja parametra w. Povećanjem vrednosti parametra w dobijaju se kraće putanje smanjenjem luka, ali se tada više ulazi u zonu opasnosti.

U Tabeli 6.11 su prikazane dužine putanja i stepeni opasnosti za putanje dobijene primenom predloženog metoda u četvrtom test primeru.

U ovom test primeru rezultate nije moguće tačno uporediti sa rezultatima u [148] zato što tačne pozicije statičkih prepreka i izvora opasnosti nisu poznate. Najkraća putanja dobijena u [148] je imala dužinu 38.2251 i ide kroz oba izvora opasnosti, tako

Tabela 6.11 Dužine putanja i stepeni opasnosti za četvrti test primer kada se koriste različite vrednosti parametra w

	L(PH)			D(PH)		
w	mean	std	best	mean	std	best
0.0	91.809	22.931	76.052	0.050	0.008	0.048
0.1	69.197	5.970	65.146	0.052	0.009	0.048
0.2	64.225	5.100	61.232	0.052	0.009	0.048
0.3	61.242	3.900	57.964	0.281	0.002	0.278
0.4	57.417	8.352	49.019	0.296	0.281	0.019
0.5	56.825	6.607	43.897	0.310	0.101	0.164
0.6	52.077	6.892	48.923	0.337	0.216	0.121
0.7	51.126	3.772	48.032	0.383	0.127	0.297
0.8	49.132	3.851	46.221	0.523	0.132	0.449
0.9	42.059	3.593	39.112	0.821	0.216	0.716
1.0	38.584	0.000	38.584	0.998	0.000	0.998

da je stepen rizika u [148] bio između 0.9087 i 1. Naš predloženi algoritam je našao sličnu najkraću putanju i izračunati stepen opasnosti je bio 0.998 (plava putanja na Slici 6.7). Na osnovu rezultata dobijenih u ovom složenom primeru ponovo se može doći do zaključka da naš metod pravi dobar Pareto front. Dužina putanje se smanjuje sa povećanjem vrednosti parametra w, a stepen opasnosti se smanjuje zajedno sa w.

Kvalitet i robustnost predloženog algoritma je dokazana vrednostima standardne devijacije koja je u većini slučajeva veoma mala ili jednaka nuli. U nekim slučajevima, velika standardna devijacija je zbog toga što funkcija cilja može biti poboljšana bilo skraćenjem putanje ili izbegavanjem zona opasnosti. U nekim konfiguracijama prepreka i zona opasnosti moguće je imati dva nedominirana rešenja skoro istog kvaliteta (iste vrednosti funkcije cilja), ali veoma različita. Jedno rešenje može biti kratko, ali opasno, a drugo bezbedno ali dugačko. U ponovljenjim izvršavanjima će dobra rešenja, u smislu ciljne funkcije, uvek biti nađena, ali ova rešenja će biti kombinacija pomenutih rešenja, što povećava varijansu.

6.3 Planiranje putanje robota primenom poboljšanog brain storm algoritma

6.3.1 Matematički model

Zbog velikog broja različitih ciljeva, osobina sredine i uslova u okruženju, u istraživačkim radovima se predlažu veoma različiti modeli putanje, razne optimizacione metode i uključuju razna ograničenja [183], [195], [196].

U ovom istraživanju se razmatra dvodimenzionalno planiranje putanje mobilnog robota u grid-baziranom okruženju sa statičkim preprekama. Model okruženja koji se koristi u ovom istraživanju je veoma često korišćen [109], [197].

Oblast od interesa, tj. prostor pretraživanja, je pravougaoni prostor dužine x_{max} i visine y_{max}. Prostor je podeljen u kvadratne ćelije jednake veličine, grid ćelije. Ako inicijalna oblast od interesa nema pravougaoni oblik, ona se može proširiti do pogodnog oblika i dodate ćelije se tada smatraju preprekama. Prepreke O_j, gde je $j = 1, 2, \ldots, M$ su postavljene unutar prostora pretraživanja. Svaka prepreka je definisana preko jedne ili više grid ćelija. Okruženje je implementirano kao matrica dimenzija $x_{max} \times y_{max}$, gde nule u matrici predstavljaju slobodan prostor, a jedinice predstavljaju prepreke. Na primer, sledeća matrica predstavlja okruženje prikazano Slikom 6.8.

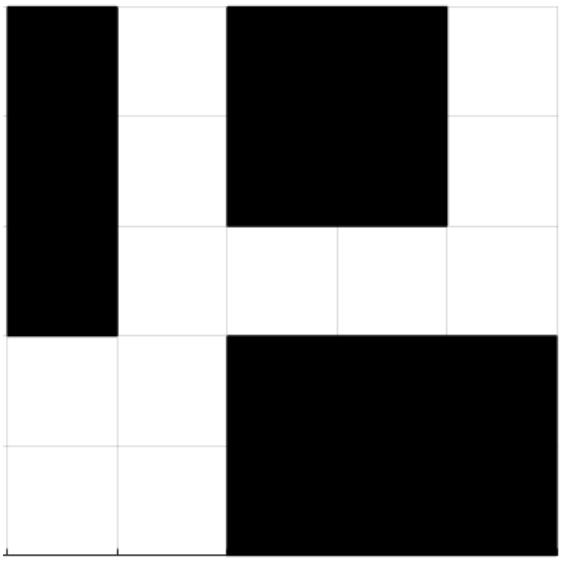
$$E = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \tag{6.19}$$

Putanja PH je definisana preko početne tačke S, ciljne tačke T i n_{path} tačaka između njih, označenih kao $ph_1, ph_2, \ldots, ph_{n_{path}}$:

$$PH = (S, ph_1, ph_2, \ldots, ph_{n_{path}}, T) \tag{6.20}$$

Svaka tačka je definisana svojim grid koordinatama (x, y), tj. $x_{min} \leq x \in N \leq x_{max}$ i $y_{min} \leq y \in N \leq y_{max}$. Centar ćelije grida se smatra grid tačkom.

Dužina putanje se definiše kao suma rastojanja između dve susedne tačke na pu-



Slika 6.8 Okruženje definisano jednačinom 6.19

tanji:

$$L(PH) = \sum_{i=0}^{n} d(ph_i, ph_{i+1}) \tag{6.21}$$

gde je $d(a, b)$ euklidsko rastojanje između a i b:

$$d(a, b) = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2} \tag{6.22}$$

6.3.2 Predloženi algoritam

Naš predloženi metod kombinuje proceduru lokalnog pretraživanja za nalaženje najkraće putanje u grafu sa brain storm optimizacionim algoritmom.

Prvi korak u nalaženju optimalne putanje je da se nađe izvodljiva putanja zato što neizvodljive putanje nisu od koristi u realnim situacijama. U cilju poboljšanja šansi za nalaženjem optimalne putanje pomoću BSO algoritma, koristili smo ideju generisanja izvodljivih putanja slično kao što je to predloženo u [196].

U [196], putanja se formira na inkrementalan način, tako da se prvo generiše N slučajnih tačaka (sve u slobodnom prostoru, van prepreka). Putanja se zatim formira polazeći od startne tačke S, a sledeća tačka se bira iz N generisanih slučajnih tačaka. Koristi se najbliža tačka kojom se formira izvodljivo rešenje. Nakon toga se izvršava ista procedura za izabrane tačke, dok se ne dostigne ciljna tačka. Ovo je u suštini

Dijkstra algoritam za nalaženje najkraće putanje u grafu čiji su čvorovi startna tačka, ciljna tačka i slučajno generisane tačke, a grane su njihove veze. U [196] se grane koje presecaju prepreke ne razmatraju.

U ovom istraživanju smo koristili metod lokalnog pretraživanja za generisanje inicijalne populacije brain storm algoritma. Predloženi algoritam počinje rad tako što generiše N grid tačaka u slobodnom prostoru (van prepreka). Ove tačke se smatraju čvorovima grafa. Računa se euklidsko rastojanja između svakog para tačaka i formira se matrica rastojanja. Kako bismo obezbedili da samo izvodljiva rešenja mogu biti nađena kao optimalna putanja, podesili smo da rastojanja između tačaka čije veze vode do generisanja neizvodljivih rešenja budu beskonačna. Procedura lokalnog pretraživanja slična onoj u [196] je korišćena da bi se našla najkraća putanja u tom grafu gde su čvorovi generisane tačke, zajedno sa startnom i ciljnom tačkom. Svaka tačka je direktno povezana sa svakom drugom tačkom. S obzirom da neizvodljive grane imaju beskonačnu dužinu, one neće biti deo najkraće putanje.

S obzirom da su tačke generisane na slučajan način, u slučaju složenog okruženja sa velikim brojem prepreka se može desiti da izvodljiva putanja dobijena povezivanjem generisanih tačaka ne postoji. Ova se verovatnoća smanjuje sa povećanjem broja generisanih tačaka, ali se tada složenost povećava. U takvim slučajevima, ako je dužina najkraće putanje nađene primenom metode lokalnog pretraživanja beskonačna, mi generišemo novi skup slučajnih tačaka i ponavljamo proceduru dok se ne pronađe izvodljivo rešenje.

Opisani metod je korišćen za postavljanje inicijalne populacije za BSO algoritam, tako da se on ponavlja n puta, gde je n veličina populacije. Dimenzija problema je jednaka vrednosti $2 * n_{path}$, gde je n_{path} broj tačaka putanje između početne i ciljne tačke, i množi se sa dva zato što su tačke definisane u dvodimenzionalnom prostoru. Rešenja dobijena primenom algoritma lokalnog pretraživanja nemaju nužno istu veličinu. U cilju obezbeđivanja da sva rešenja imaju istu dimenziju, prvo se nalazi rešenje koje ima najveću dimenziju. Rešenja koja imaju manje tačaka se proširuju deljenjem najdužeg segmenta putanje u onoliko jednakih delova koliko je neophodno.

Kada su primenom predloženog algoritma lokalnog pretraživanja nađena inicijalna rešenja i nakon što su njihove dimenzije podešene i prilagođene, primenjuje se BSO algoritam kako bi se rešenja optimizovala. Dobijena izvodljiva rešenja se koriste umesto

slučajnih rešenja, kao prva generacija BSO algoritma. S obzirom da je predloženo rešenje bazirano na gridu, a brain storm algoritam radi sa realnim brojevima, izvršili smo zaokruživanje svake dobijene koordinate na najbližu celobrojnu vrednost.

Rešenja dobijena primenom BSO algoritma su evaluirana primenom sledeće fitnes funkcije. Cilj predloženog metoda je nalaženje najkraće izvodljive putanje. Obezbedili smo da inicijalna populacija BSO algoritma sadrži samo izvodljiva rešenja, ali u toku procesa pretraživanja neka rešenja mogu biti promenjena tako da putanja prolazi kroz prepreku. U nekim radovima su predložene metode korekcije rešenja da bi se dobila izvodljiva rešenja koja su najsličnija generisanim. U ovom istraživanju se dozvoljavaju neizvodljiva rešenja, ali je to prikazano kroz odgovarajuću vrednost fitnes funkcije. Ako su generisana rešenja neizvodljiva, na vrednost fitnes funkcije se dodaje vrednost kazne (penalty). Dužina putanje $L(PH)$ je određena jednačinom 6.21, i fitnes funkcija se izračunava na sledeći način:

$$fitness(PH) = L(PH) + penalty \tag{6.23}$$

gde je

$$penalty = \begin{cases} \frac{L(PH)}{2}, & \text{ako putanja nije izvodljiva} \\ 0, & \text{u suprotnom} \end{cases} \tag{6.24}$$

Uvođenjem *penalty* vrednosti u fitnes funkciju, BSO će na kraju odbaciti neizvodljiva rešenja zato što će ona imati veću vrednost fitnes funkcije u poređenju sa sličnim izvodljivim rešenjima. Predloženi metod za planiranje putanje mobilnih robota u statičkom okruženju je dat u Algoritmu 6.1.

6.3.3 Rezultati simulacije

Predloženi brain storm optimizacioni algoritam kombinovan sa algoritmom lokalnog pretraživanja, primenjen za planiranje putanje mobilnih robota je implementiran korišćenjem programskog paketa Matlab R2016a. Eksperimenti su obavljeni na računarskoj platformi: Intel Core i7-3770K CPU na 4GHz, 8GB RAM, Windows 10 Professional OS.

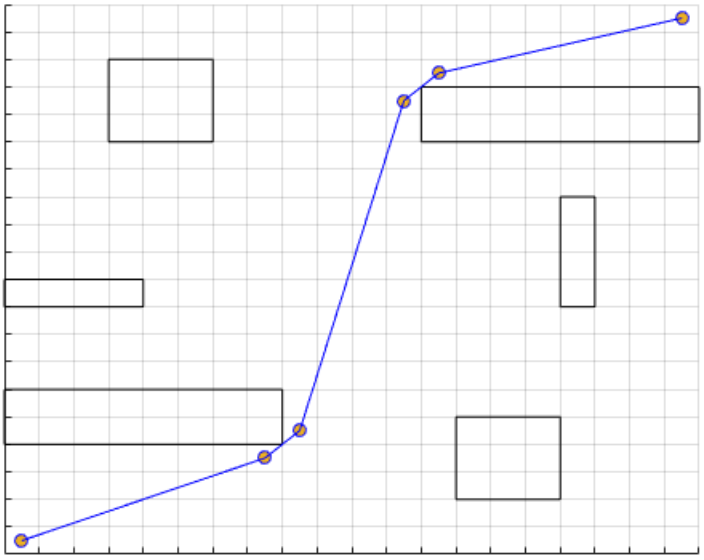
Parametri predloženog metoda su podešeni na empirijski način, pomoću prethodnih testova. Za nalaženje izvodljivih putanja pomoću procedure lokalnog pre-

Algoritam 6.1: Pseudokod predloženog metoda za problem planiranja putanje

```
Inicijalizacija primenom procedure lokalnog pretraživanja;
for i=1 to n do
    repeat
        Generisati N slučajnih tačaka u ćelijama grida koje ne sadrže prepreke;
        Izračunati matricu rastojanja između generisanih tačaka, početne i ciljne tačke, podešavajući
          rastojanje za neizvodljive segmente kao beskonačno;
        Naći najkraću putanju primenom lokalnog pretraživanja, a na osnovu matrice rastojanja;
    until najkraća putanja je manja od beskonačnosti;
end
Naći najveću dimenziju rešenja, d;
for all rešenja s_i sa dimenzijom manjom od d do
    Naći najduži segment putanje od s_i i podeliti ga na jednake delove pomoću d − dim(s_i) tačaka;
end
repeat
    Podeliti n rešenja u m klastera primenom k-srednjeg algoritma;
    Rangirati rešenja u svakom klasteru i označiti najbolja rešenja kao centre klastera;
    Metodom slučajnog izbora generisati vrednost r između 0 i 1;
    if r < p_5a then
        Metodom slučajnog izbora odabrati centar klastera;
        Metodom slučajnog izbora generisati individuu koja će da zameni izabrani centar klastera;
    end
    repeat
        Generisati nova rešenja;
        Metodom slučajnog izbora generisati vrednost r između 0 i 1;
        if r < p_6b then
            Metodom slučajnog izbora izabrati klaster sa verovatnoćom p_6bi;
            Metodom slučajnog izbora generisati vrednost r_1 između 0 i 1;
            if r_1 < p_6bii then
                Izabrati centar klastera i dodati mu slučajnu vrednost da bi se generisala nova individua;
            else
                Metodom slučajnog izbora izabrati rešenje iz izabranog klastera i dodati mu slučajnu
                  vrednost da bi se generisalo novo;
            end
        else
            Metodom slučajnog izbora izabrati dva klastera;
            Generisati slučajnu vrednost r_2 između 0 i 1;
            if r_2 < p_6c then
                Dva centra klastera se kombinuju, da bi se generisalo novo rešenje;
            else
                Dve individue iz svakog od izabranih klastera se biraju metodom slučajnog izbora i
                  zatim se kombinuju da bi se generisala nova individua;
            end
        end
        Novo generisano rešenje se poredi sa postojećim rešenjem sa istim individualnim indeksom, i bolje
          se zadržava;
    until n novih individua je generisano;
until dostignut maksimalni broj iteracija;
return najbolje rešenje u populaciji
```

traživanja, generisano je 50 slučajnih tačaka ($N = 50$). Veličina populacije n BSO algoritma je postavljena na 20, a broj klastera m je 4. Verovatnoća slučajne promene rešenja, p_{5a}, je postavljena na 0.2. Vrednosti verovatnoća p_{6b}, p_{6bii} i p_{6c} su 0.8, 0.4 i 0.5, respektivno. Maksimalan broj iteracija je bio 500.

Model za problem planiranja putanje baziran na gridu je razmatran u [109]. Radi



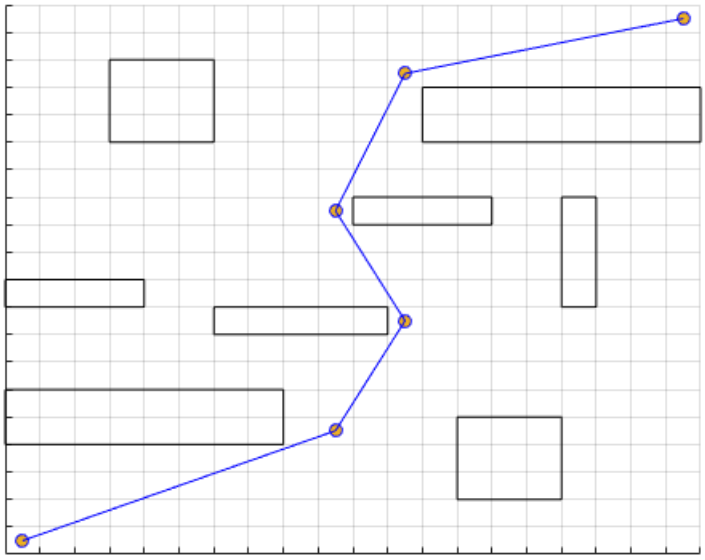
Slika 6.9 Prvo test okruženje, 6 prepreka

rešavanja problema planiranja putanje predložen je membranom inpirisani algoritam baziran na optimizaciji rojevima čestica (mPSO). Testirali smo naš predloženi BSO, kombinovan sa procedurom lokalnog pretraživanja, u tri okruženja koja su korišćena u [109]. Prostor pretraživanja je bio grid dimenzija 20×20, sa 6, 8 i 10 prepreka. Veličina populacije je bila 100 i maksimalni broj iteracija je bio 2000, što je značajno veće od vrednosti parametara u našem predloženom algoritmu. Naš predloženi metod je izvršavan 30 puta, za svako od test okruženja, i ista putanja je nađena u svakom izvršavanju za prva dva test primera, sa 6 i 8 prepreka, dok su za okruženje sa 10 prepreka pronađene dve različite putanje u 30 izvršavanja. Ovo dokazuje robustnost brain storm optimizacionog algoritma kombinovanog sa Dijkstra algoritmom.

Prvo test okruženje sa 6 prepreka, kao i dobijena putanja primenom predloženog metoda, su prikazani na Slici 6.9. Početna i ciljna tačka su postavljene u svim okruženjima u donji levi i gornji desni ugao, respektivno. Putanja koja je nađena primenom našeg predloženog metoda je bila ista kao putanja prikazana u [109]. Ona sadrži 4 tačke između početka i cilja. Ove četiri tačke su potrebne radi izbegavanja dve prepreke koje ne dozvoljavaju da se ide pravom linijom od početne do ciljne tačke.

Drugo test okruženje je slično prethodnom, sa dodatkom dve prepeke. Da bi se od početke tačke došlo do ciljne tačke, putanja mora da zaobiđe ove dodate prepreke. Svi modeli putanje koji podrazumevaju kretanje unapred u svakom koraku nisu u

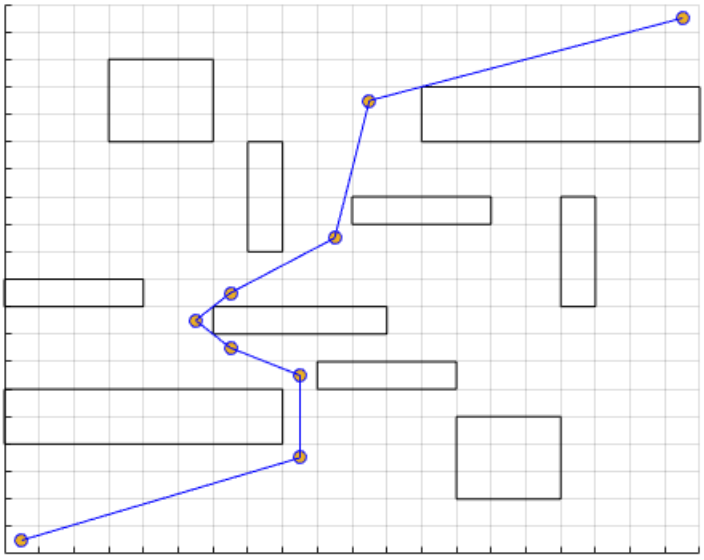
mogućnosti da nađu izvodljivo rešenje za ovo okruženje. Putanja nađena primenom našeg predloženog BSO metoda je prikazana na Slici 6.10. Putanja takođe ima četiri tačke između početne i krajnje tačke, kao u prethodnom primeru. U [109], mPSO je našao putanju sa 6 tačaka između.



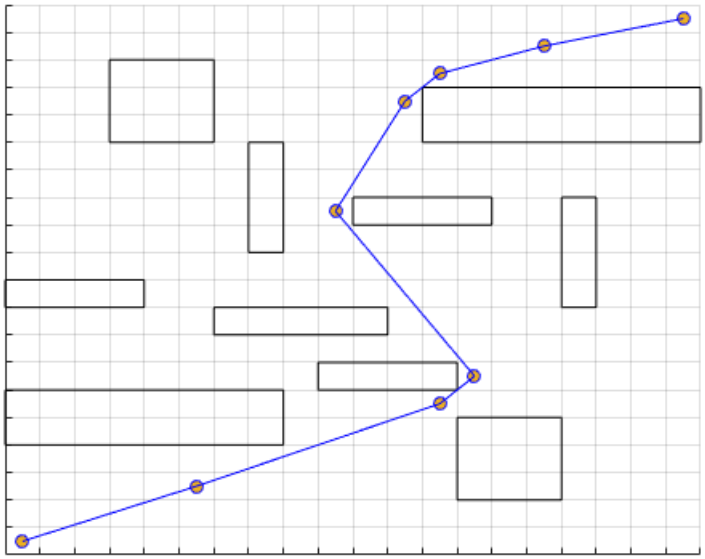
Slika 6.10 Drugo test okruženje, 8 prepreka

Treće test okruženje je najkomplikovanije. U prethodni test primer su dodate dve nove prepreke. Na Slici 6.11 je prikazano test okruženje sa pronađenim rešenjem. Optimalna putanja za ovaj primer napravljena je pomoću 7 tačaka, koje su neophodne kako bi se prepreke zaobišle tačno oko njihovih ivica. Na Slici 6.12 je prikazano drugo rešenje koje je dobijeno primenom našeg metoda. Dužina putanje za ovo rešenje je bila 35.1172, dok je dužina putanje prvog rešenja, prikazanog na Slici 6.11, bila 34.7999.

Dodatno, izvršili smo poređenje našeg predloženog metoda sa metodom predoženim u [197], gde su za problem planiranja putanje predložene dve metode poboljšanog algoritma optimizacije rojevima čestica. U prvoj varijanti, nelinearni koeficijenti sa inercijskim težinama su primenjeni u PSO kako bi se kontrolisala lokalna i globalna sposobnost pretraživanja. Drugi predloženi metod je kombinovao PSO sa simuliranim kaljenjem, koje sprečava PSO da se zaglavi u lokalnom optimumu. Predloženi metod je testiran u dva različita okruženja. Veličina oblasti je bila 20×20, a prvo okruženje je sadržalo 9 prepreka, dok je drugo okruženje bilo složenije i sadržalo je 17 statičkih prepreka. Predložene PSO metode su upoređivane sa originalnim PSO.

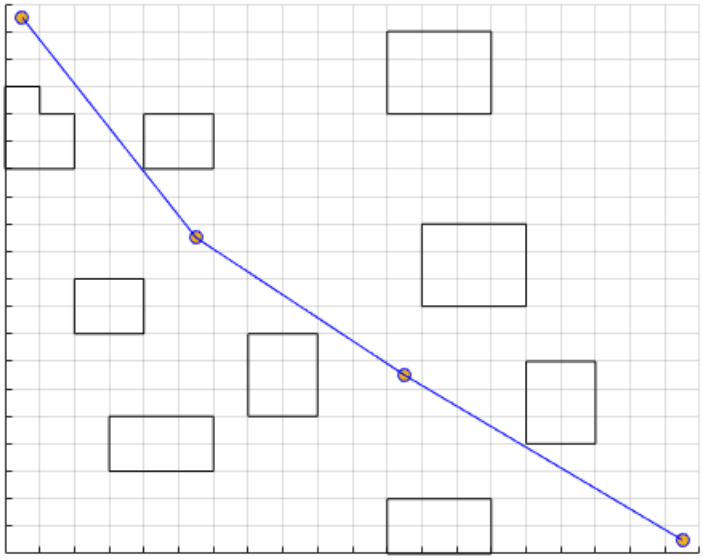


Slika 6.11 Treće test okruženje, 10 prepreka



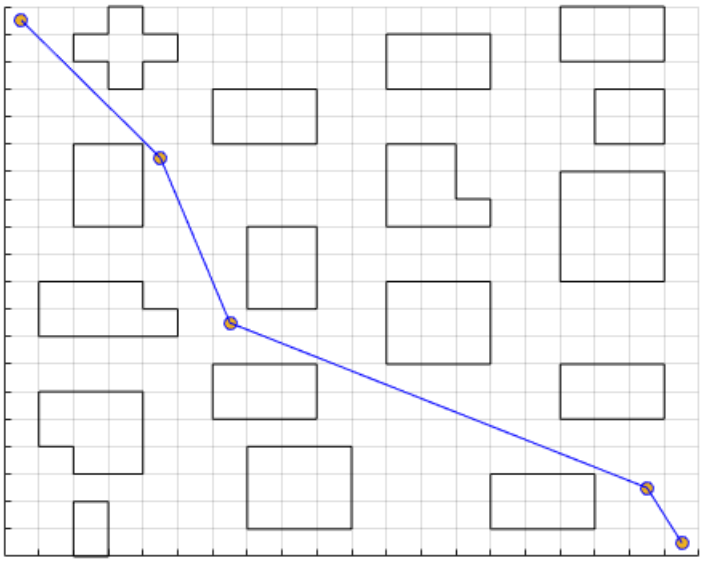
Slika 6.12 Treće test okruženje, 10 prepreka

Za slučaj prvog okruženja u [197] (četvrto u ovom istraživanju), osnovni PSO algoritam je dobio rešenje gde je dužina putanje bila 28.5006, PSO sa nelinarnim inercijskim težinama je našao putanju dužine 27.6853, dok je PSO sa simuliranim kaljenjem dobio rešenje čija je dužina 28.1973. Naš predloženi BSO metod je našao putanju kraću u poređenju sa sva tri prethodna metoda, i dužina joj je 27.2442. Dobijeno rešenje je prikazano na Slici 6.13.



Slika 6.13 Četvrto test okruženje, 9 prepreka

Drugi test izvršen u [197] predstavlja pretragu za putanjom u prilično složenom okruženju sa 17 statičkih prepreka u gridu dimenzija 20×20. Test okruženje, zajedno sa putanjom dobijenom primenom našeg predloženog algoritma je prikazano na Slici 6.14. Dužina putanje dobijene primenom BSO metoda je bila 28.2802. Izračunali



Slika 6.14 Peto test okruženje, 17 prepreka

smo dužine putanja za putanje date u [197] i vrednosti su bile 32.0153, 30.3623 i

28.7831, dobijene primenom osnovnog PSO, PSO sa nelinearnim inercijskim težinama i PSO sa simuliranim kaljenjem, respektivno (što je različito od dužine putanja koje su prikazane na slikama). Ponovo je naš predloženi metod našao bolje rešenje u poređenju sa sve tri varijante PSO algoritma. Opisani rezultati su prikazani u Tabeli 6.12.

Tabela 6.12 Dužine putanja dobijenih primenom PSO, PSO sa nelinearnim inercijskim težinama (NLI-PSO) i PSO sa simuliranim kaljenjem (SA-PSO) [197], i predloženog BSO metoda

Okruženje	PSO	NLI-PSO	SA-PSO	BSO
Test 4	28.5006	27.6853	28.1973	**27.2442**
Test 5	32.0153	30.3623	28.7831	**28.3802**

Zaključak

U ovoj disertaciji smo analizirali načine rešavanja problema planiranja putanje robota primenom prirodom inspirisanih metaheuristika, posebno posvećujući pažnju algoritmima inteligencije rojeva. Prilagodili smo i primenili brain storm optimizacioni algoritam na rešavanje problema planiranja putanje robota.

U prvom delu istraživanja smo primenili osnovnu verziju BSO algoritma na problem planiranje putanje UCAV letelica. Potrošnja goriva i bezbednost su razmatrani kao kriterijumi optimalnosti putanje. Predloženi metod je testiran u test okruženju iz literature koji podrazumeva kružno predstavljanje zona opasnosti i različite nivoe pretnje. Naš predloženi metod je bio upoređivan sa deset drugih metoda iz literature. Na osnovu rezultata simulacije se može zaključiti da predloženi brain storm optimizacioni algoritam ispoljava veoma obećavajuće osobine. On je pokazao bolje performanse za manje dimenzije problema, dok je za veće dimenzije bilo potrebno više iteracija, mada su rezultati i u tom slučaju pokazali poboljšanje u poređenju sa ostalim testiranim algoritmima.

U drugom delu istraživanja smo primenili osnovni brain storm optimizacioni algoritam na problem planiranja putanje robota u neizvesnom okruženju sa statičkim preprekama. Predložili smo probabilistički model za određivanje stepena opasnosti, za izvore opasnosti sa nepoznatim tačnim pozicijama. Dva kontradiktorna kriterijuma, dužina putanje i bezbednost, su povezana uvođenjem kontrolnog parametra u funkciju cilja. Predloženi metod rešava problem neizvodljivih rešenja dodavanjem vrednosti kazne u funkciju cilja. Kombinacija kazne i povećane eksploracije je obezbedila da se uvek generišu izvodljiva rešenja. Poređenjem sa konkurentnim algoritmom je dokazano da je naš predloženi metod, iako jednostavniji, efikasniji i robustniji zato što su dobijena bolja rešenja u svim slučajevima.

Na kraju je razmatran problem planiranja putanje mobilnog robota u dvodimenzionalnom okruženju primenom poboljšanog BSO algoritma. Predloženi metod kombinuje BSO algoritam sa metodom lokalnog pretraživanja da bi se našla najkraća putanja u grafu, u cilju pretraživanja optimalne putanje u okruženju sa statičkim preprekama. Dužina putanje je korišćena kao jedini cilj. Inicijalna izvodljiva rešenja za BSO su generisana primenom determinističke procedure lokalnog pretraživanja, a BSO je korišćen da bi se dalje optimizovala putanja. Predloženi metod je testiran za

pet različitih scenarija i dokazano je da može da nađe optimalna izvodljiva rešenja.

Buduća istraživanja mogu uključivati hibridizaciju i modifikaciju brain storm optimizacionog algoritma kako bi se poboljšala brzina konvergencije i algoritam prilagodio rešavanju problema sa većim dimenzijama. S obzirom da smo u našem istraživanju za opis problema planiranja putanje koristili dvodimenzionalni prostor, buduće istraživanje može da uključi i treću dimenziju, tj. UCAV altitudu. BSO algoritam je pokazao robustnost i superiorne performanse u testiranim slučajevima. Međutim, njegov potencijal je još i veći imajući u vidu njegove jedinstvene osobine klasterovanja rezultata koje mogu da se iskoriste za složenije, višeciljne formulacije problema planiranja putanje UCAV letelica. Trenutno su aktuelna istraživanja koja imaju za cilj rešavanje problema samoadaptivnog planiranja putanje za jednu UCAV letelicu, kao i kolaborativno planiranje putanje, ako je u pitanju flota od više letelica. Adaptivno planiranje putanje treba da ima mogućnost analize podataka u realnom vremenu u slučaju neodređenog i dinamičnog okruženja i replaniranja putanje. Izazovi kolaborativnog planiranja putanje se odnose na koordinaciju između više UCAV letelica, uključujući formaciju leta, ograničenja dolaska na cilj, izbegavanja konflikata i sl.

Buduća istraživanja mogu da uključe i brojna poboljšanja. S obzirom da putanje nisu samo skupovi tačaka, već imaju brojne inherentne relacije i zavisnosti, tu činjenicu je moguće iskoristiti kako bi se napravili efikasniji algoritmi. U daljem istraživanju, umesto modela okruženja baziranog na gridu, mogu se koristiti realni prostori pretraživanja, a inicijalne tačke se mogu dobiti korišćenjem određenih smernica umesto korišćenjem tačaka slučajno raspoređenih u prostoru pretraživanja.

Literatura

[1] P. Pedregal, Introduction to Optimization, Springer-Verlag, New York, 2004.

[2] XS. Yang, Nature-Inspired Metaheuristic Algorithms: Second Edition, Luniver Press, UK, 2010.

[3] E. Chong, S. Zak, An Introduction to Optimization, John Wiley and Sons, USA, 2001.

[4] B. Korte, J. Vygen, Combinatorial Optimization, Theory and Algorithms, Third Edition, Springer-Verlag, 2005.

[5] E. L. Lawler, Combinatorial Optimization:Networks and Matroids, Holt, Rinehart and Winston, USA, 1976.

[6] M. Gestal, M.P. Gomez-Carracedo, Finding Multiple Solutions with GA in Multimodal Problems, Encyclopedia of Artificial Intelligence, pp. 647-653, doi: 10.4018/9781599048499.ch098, 2009.

[7] T. H. Cormen, C. E. Leiserson,R. L. Rivest, C. Stein, Introduction to Algorithms, Second Edition, The MIT Press, USA, 2001.

[8] O. Bozorg-Haddad (ed.), Advanced Optimization by Nature-inspired Algorithms, Springer Nature Singapore Pte Ltd, Singapore, 2018.

[9] M. Jamil, XS. Yang, A literature survey of benchmark functions for global optimization problems, Int. Journal of Mathematical Modelling and Numerical Optimisation, Vol. 4, No. 2, pp. 150–194, 2013.

[10] Fister Jr., XS. Yang, I. Fister, J. Brest, D. Fister, A Brief Review of Nature-Inspired Algorithms for Optimization, Elektrotehniški Vestnik, Vol. 80, No. 3, pp. 1–7, 2013.

[11] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, M. Protasi, Complexity and Approximation, Combinatorial optimization problems and their approximability properties, Springer- Verlag, Germany, 1999.

[12] T. El-Ghazali, Metaheuristics: From Design to Implementation, John Wiley and Sons, USA, 2009.

[13] I. Boussaid, J. Lepagnot, P. Siarry, A survey on optimization metaheuristics, Information Sciences, Vol. 237, pp. 82-117, 2013.

[14] T.O. Ting, XS. Yang, S. Cheng, K. Huang, Hybrid Metaheuristic Algorithms: Past, Present, and Future. In: Yang XS. (ed.) Recent Advances in Swarm Intelligence and Evolutionary Computation. Studies in Computational Intelligence, Vol. 585, Springer, Cham, Switzerland, 2015.

[15] A. Colorni, M. Dorigo, F. Maoli, V. Maniezzo, G. Righini, M. Trubian, Heuristics from nature for hard combinatorial optimization problems, International Transactions in Operational Research, Vol. 3, No. 1, pp. 1-21, 1996.

[16] XS. Yang (ed.), Nature-Inspired Computation in Engineering, Springer International Publishing, Switzerland, 2016.

[17] XS. Yang, Nature-Inspired Optimization Algorithms, Elsevier, USA, 2014.

[18] K. L. Du, M. N. S. Swamy, Search and Optimization by Metaheuristics, Techniques and Algorithms Inspired by Nature, Springer International Publishing, Switzerland, 2016.

[19] XS. Yang, X. He, Swarm Intelligence and Evolutionary Computation: Overview and Analysis. In: XS. Yang (eds.) Recent Advances in Swarm Intelligence and Evolutionary Computation. Studies in Computational Intelligence, Vol. 585, Springer, Cham, Switzerland, 2015.

[20] S. Kirkpatrick, C. D. Gelatt, Jr., M. P. Vecchi, Optimization by Simulated Annealing, Science, Vol. 220, Issue 4598, pp. 671-680, 1983.

[21] E. Aarts, J. Korst, W. Michiels, Simulated Annealing. In: Burke E.K., Kendall G. (eds.) Search Methodologies, Springer, Boston, MA, 2005.

[22] B Suman, P. Kumar, A survey of simulated annealing as a tool for single and multiobjective optimization, Journal of the Operational Research Society, Vol. 57, No. 10, pp. 1143–1160, 2006.

[23] M. Melanie, An Introduction to Genetic Algorithms, The MIT Press, USA, 1999.

[24] M. Srinivas, L. M. Patnaik, Genetic algorithms: a survey, Computer, Vol. 27, No. 6, pp. 17-26, 1994.

[25] R. C. Eberhart, Y. Shi, Comparison between Genetic Algorithms and Particle Swarm Optimization, Proceedings of the 7th International Conference on Evolutionary Programming, pp. 611-616, San Diego, California, USA, 1998.

[26] S. Koziel, XS. Yang (eds.), Computational Optimization, Methods and Algorithms, Springer-Verlag, Germany, 2011.

[27] R. Storn, K. Price, Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, Journal of Global Optimization, Vol. 11, pp. 341–359, 1997.

[28] A.K. Qin, P.N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, 2005 IEEE Congress on Evolutionary Computation, pp. 1785-1791, Vol. 2., Scotland, UK, 2005.

[29] F. Neri, V. Tirronen, Recent advances in differential evolution: a survey and experimental analysis, Artificial Intelligence Review, Vol. 33, Issue 1–2, pp. 61–106, 2010.

[30] S. Das, P. N. Suganthan, Differential Evolution: A Survey of the State-of-the-Art, IEEE Transactions on Evolutionary Computation, Vol. 15, Issue 1, pp. 4-31, 2011.

[31] E. Dolicanin, I. Fetahovic, Monte Carlo Optimization of Redundancy of Nanotechnology Computer Memories in the Conditions of Background Radiation, Nuclear Technology and Radiation Protection, Vol. XXXIII, No. 2, 2018.

[32] L. Fisher, The Perfect Swarm: The Science of Complexity in Everday Life, Basic Books, USA, 2009.

[33] E.F. Keller, Organisms, Machines, and Thunderstorms: A History of Self-Organization, Part Two: Complexity, Emergence, and Stable Attractors, Historical Studies in the Natural Sciences, Vol. 39, No. 1, pp. 1–31, 2009.

[34] E. Bonabeau, M. Dorigo,G. Theraulaz, Swarm intelligence: from natural to artificial systems, Oxford University Press Inc, New York, NY, USA , 1999.

[35] Y. Shi, Developmental swarm intelligence: developmental learning perspective of swarm intelligence algorithms, Int J Swarm Intell Res, Vol. 5, No. 1, pp. 36–54, 2014.

[36] M. Dorigo, G. Di Caro, L. M. Gambardella, Ant Algorithms for Discrete Optimization, Artificial Life, Vol. 5, No. 2, pp. 137-172, 1999.

[37] M. Dorigo, T. Stützle, The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances. In: F. Glover, G.A. Kochenberger (eds) Handbook of Metaheuristics. International Series in Operations Research and Management Science, Vol 57. Springer, Boston, MA, USA, 2003.

[38] M. Dorigo, G. Di Caro, Ant Colony Optimization: A New Meta-Heuristic, Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), pp. 1477 , Vol. 2., Washington, DC, USA, 1999.

[39] M. Dorigo, C. Blum, Ant colony optimization theory: A survey, Theoretical Computer Science, Vol. 344, Issues 2–3, pp. 243-278, 2005.

[40] M. Gendreau, JY. Potvin (eds.), Handbook of Metaheuristics Second Edition, Springer, USA, 2010. M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), Vol. 26, No. 1, pp. 29-41, 1996.

[41] C. Blum, Ant colony optimization: Introduction and recent trends, Physics of Life Reviews, Vol. 2, Issue 4, pp. 353-373, 2005.

[42] M. Dorigo, G. Di Caro, The ant colony optimization meta-heuristic. In: D. Corne, M. Dorigo, F. Glover, (eds.) New Ideas in Optimization, pp. 11–32, McGraw Hill, London, UK, 1999.

[43] C.W. Reynolds, Flocks, herds, and schools: a distributed behavioral model, Comput Graph 21(4):25–34, 1987.

[44] F. Heppner, U. Grenander, A stochastic nonlinear model for coordinated bird flocks, In: S. Krasner (ed.), The Ubiquity of Chaos, AAAS Publications, Washington DC, USA, 1990.

[45] J. Kennedy, R.C. Eberhart, Particle swarm optimization, Proceedings of the IEEE international conference on neural networks, pp. 1942–1948, Perth, Australia, 1995.

[46] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, Proceedings of the IEEE international conference on evolutionary computation, pp. 69–73, Anchorage, Alaska, USA, 1998.

[47] Y. Zhang, S. Wang, G. Ji, A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications, Mathematical Problems in Engineering, Vol. 2015, Article ID 931256, 38 pages, 2015.

[48] T. D. Seeley, The wisdom of the hive : the social physiology of honey bee colonies, Harvard University Press, Cambridge, Massachusetts, USA, 1995.

[49] D. Karaboga, B. Gorkemli, C. Ozturk, N. Karaboga, A comprehensive survey: artificial bee colony (ABC) algorithm and applications, Artificial Intelligence Review, Vol. 42, Issue 1, pp. 21–57, 2014.

[50] J. C. Bansal, H. Sharma, S. S. Jadon, Artificial bee colony algorithm: a survey, International Journal of Advanced Intelligence Paradigms, Vol. 5, Issue 1-2, pp. 123-159, 2013.

[51] F. S. Abu-Mouti, M. E. El-Hawary, Overview of Artificial Bee Colony (ABC) Algorithm and Its Applications, IEEE International Systems Conference SysCon 2012, pp. 1-6, Vancouver, Canada, 2012.

[52] D. Karaboga, B. Akay, A comparative study of Artificial Bee Colony algorithm, Applied Mathematics and Computation, Vol. 214, Issue 1, pp. 108-132, 2009.

[53] D. Karaboga, An Idea based on Honey Bee Swarm for Numerical Optimization, Techn.Rep. TR06, Erciyes Univ. Press, Erciyes, Turkey, 2005.

[54] B. Akay, D. Karaboga, A modified Artificial Bee Colony algorithm for real-parameter optimization, Information Sciences, Vol. 192, pp. 120-142, 2012.

[55] XS. Yang, Nature-Inspired Metaheuristic Algorithms, Luniver Press, UK, 2008.

[56] O. Watanabe, T. Zeugmann (eds.), Stochastic Algorithms: Foundations and Applications, Proceedings of the 5th International Symposium, SAGA 2009, Sapporo, Japan, 2009.

[57] M. Bramer, R. Ellis, M. Petridis (eds.), Research and Development in Intelligent Systems XXVI Incorporating Applications and Innovations in Intelligent Systems XVII, Springer-Verlag London, UK, 2010.

[58] XS. Yang, Engineering Optimization: An Introduction with Metaheuristic Applications, John Wiley and Sons, USA, 2010.

[59] XS. Yang (ed.), Cuckoo Search and Firefly Algorithm Theory and Applications, Springer International Publishing, Switzerland, 2014.

[60] XS. Yang, Firefly Algorithms for Multimodal Optimization. In: O. Watanabe, T. Zeugmann (eds.) Stochastic Algorithms: Foundations and Applications SAGA 2009, Lecture Notes in Computer Science, Vol. 5792, Springer, Berlin, Germany,2009.

[61] R.B. Payne, M.D. Sorenson, K. Klitz, The Cuckoos, Oxford University Press, UK, 2005.

[62] XS. Yang, S. Deb, Engineering optimisation by cuckoo search, Int. J. Mathematical Modelling and Numerical Optimisation, Vol. 1, No. 4, pp. 330-343, 2010.

[63] XS. Yang, S. Deb, Cuckoo Search via Levy Flights, 2009 World Congress on Nature and Biologically Inspired Computing (NaBIC), pp. 210-214, Coimbatore, India, 2009.

[64] XS. Yang, A New Metaheuristic Bat-Inspired Algorithm, In: J.R. González, D.A. Pelta, C. Cruz, G. Terrazas, N. Krasnogor (eds.) Nature Inspired Coo-

perative Strategies for Optimization (NICSO 2010). Studies in Computational Intelligence, Vol. 284, Springer, Berlin, Germany, 2010.

[65] A.H. Gandomi, XS. Yang, A.H. Alavi, S. Talatahari, Bat algorithm for constrained optimization tasks, Neural Computing and Applications, Vol. 22, Issue 6, pp. 1239–1255, 2013.

[66] XS. Yang, A.H. Gandomi, Bat algorithm: a novel approach for global engineering optimization, Engineering Computations, Vol. 29, Issue 5, pp. 464—483, 2012.

[67] Y. Shi, Brainstorm optimization algorithm, In: Y. Tan, Y. Shi, Y. Chai, G. Wang (eds.), Advances in swarm intelligence, lecture notes in computer science, Vol. 6728, pp. 303–309, Springer, Berlin, 2011.

[68] Y. Shi, An optimization algorithm based on brainstorming process, Int J Swarm Intell Res, Vol. 2, No. 4, pp. 35–62, 2011.

[69] Z. Cao, Y. Shi, X. Rong, B. Liu, Z. Du, B. Yang, Random Grouping Brain Storm Optimization Algorithm with a New Dynamically Changing Step Size, In: Y. Tan, Y. Shi, F. Buarque, A. Gelbukh, S. Das, A. Engelbrecht (eds.), Advances in Swarm and Computational Intelligence ICSI 2015, Lecture Notes in Computer Science, Vol. 9140, Springer, Cham, Switzerland, 2015.

[70] Z. Zhan, J. Zhang, Y. Shi, H. Liu, A modified brain storm optimization, 2012 IEEE Congress on Evolutionary Computation, pp. 1-8, Brisbane, QLD, Australia, 2012.

[71] M. El-Abd, Brain storm optimization algorithm with re-initialized ideas and adaptive step size, 2016 IEEE Congress on Evolutionary Computation (CEC), , pp. 2682-2686, Vancouver, Canada, 2016.

[72] M. El-Abd, Global-best brain storm optimization algorithm, Swarm and Evolutionary Computation, Vol. 37, pp. 27–44, 2017.

[73] D. Zhou, Y. Shi, S. Cheng, Brain Storm Optimization Algorithm with Modified Step-Size and Individual Generation, In: Y.Tan, Y.Shi, Z. Ji (eds.), Advances in

Swarm Intelligence, ICSI 2012, Lecture Notes in Computer Science, Vol. 7331, Springer, Berlin, Heidelberg, Germany, 2012.

[74] J. Xue, Y. Wu,Y. Shi, S. Cheng, Brain Storm Optimization Algorithm for Multi-objective Optimization Problems, In: Y.Tan, Y.Shi, Z. Ji (eds.), Advances in Swarm Intelligence, ICSI 2012, Lecture Notes in Computer Science, Vol. 7331, Springer, Berlin, Heidelberg, Germany, 2012.

[75] J. Chen, Y. Xie, J. Ni, Brain Storm Optimization Model Based on Uncertainty Information, 2014 Tenth International Conference on Computational Intelligence and Security, pp. 99-103, Kunming, China, 2014.

[76] J. Chen, S. Cheng,Y. Chen, Y. Xie, Y. Shi, Enhanced Brain Storm Optimization Algorithm for Wireless Sensor Networks Deployment. In: Y. Tan, Y. Shi, F. Buarque, A. Gelbukh, S. Das, A. Engelbrecht (eds) Advances in Swarm and Computational Intelligence, ICSI 2015, Lecture Notes in Computer Science, Vol. 9140, Springer, Cham, Switzerland, 2015.

[77] H. Zhu, Y. Shi, Brain storm optimization algorithms with k-medians clustering algorithms, 2015 Seventh International Conference on Advanced Computational Intelligence (ICACI), pp. 107-110, Wuyi, China, 2015.

[78] Y. Shi, Brain storm optimization algorithm in objective space, 2015 IEEE Congress on Evolutionary Computation (CEC), pp. 1227-1234, Sendai, Japan, 2015.

[79] S. Cheng, Y. Shi, Q. Qin, T. O. Ting, R. Bai, Maintaining population diversity in brain storm optimization algorithm, 2014 IEEE Congress on Evolutionary Computation (CEC), pp. 3230-3237, Beijing, China, 2014.

[80] Z. Yang, Y. Shi, Brain storm optimization with chaotic operation, 2015 Seventh International Conference on Advanced Computational Intelligence (ICACI), pp. 111-115, Wuyi, China, 2015.

[81] XS. Yang, Chaos-Enhanced Firefly Algorithm with Automatic Parameter Tuning, International Journal of Swarm Intelligence Research (IJSIR), Vol. 2, No. 4, 2011.

[82] Z. Jia, H. Duan, Hybrid brain storm optimisation and simulated annealing algorithm for continuous optimisation problems, Int. J. Bio-Inspired Computation, Vol. 8, No. 2, pp. 109-121, 2016.

[83] K.R. Krishnanand, S.M.F. Hasani, B.K. Panigrahi, S.K. Panda, Optimal Power Flow Solution Using Self–Evolving Brain–Storming Inclusive Teaching–Learning–Based Algorithm, In: Y. Tan, Y. Shi, H. Mo (eds) Advances in Swarm Intelligence, ICSI 2013, Lecture Notes in Computer Science, Vol. 7928, Springer, Berlin, Heidelberg, Germany, 2013.

[84] Z. Cao, X. Hei, L. Wang, Y. Shi, X. Rong, An Improved Brain Storm Optimization with Differential Evolution Strategy for Applications of ANNs, Mathematical Problems in Engineering, Vol. 2015, Article ID 923698, 18 pages, 2015.

[85] Y. Wu, L. Xie, Q. Liu, Multi-objective Brain Storm Optimization Based on Estimating in Knee Region and Clustering in Objective-Space, In: Y. Tan, Y. Shi, B. Niu (eds.), Advances in Swarm Intelligence, 7th International Conference, ICSI 2016, Bali, Indonesia, Lecture Notes in Computer Science, Vol. 9713, Springer, Berlin, Heidelberg, Germany, 2016.

[86] L. Xie, Y. Wu, A Modified Multi-Objective Optimization Based on Brain Storm Optimization Algorithm. In: Y. Tan, Y. Shi, C.A.C. Coello (eds), Advances in Swarm Intelligence, ICSI 2014, Lecture Notes in Computer Science, Vol. 8795, Springer, Cham, Switzerland, 2014.

[87] M. Ester, H.P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, Proceedings of the Second International Conference on Knowledge Discovery and Data Mining KDD'96, pp. 226-231, Portland, Oregon, USA, 1996.

[88] X. Guo, Y. Wu, L. Xie, Modified Brain Storm Optimization Algorithm for Multimodal Optimization In: Y. Tan, Y. Shi, C.A.C. Coello (eds), Advances in Swarm Intelligence, ICSI 2014, Lecture Notes in Computer Science, Vol. 8795, Springer, Cham, Switzerland, 2014.

[89] Z. Zhan, W. Chen, Y. Lin, Y. Gong, Y. Li, J. Zhang, Parameter investigation in brain storm optimization,2013 IEEE Symposium on Swarm Intelligence (SIS), pp. 103-110, Singapore, 2013.

[90] S. Cheng, Y. Shi, Q. Qin, S. Gao, Solution clustering analysis in brain storm optimization algorithm, 2013 IEEE Symposium on Swarm Intelligence (SIS), pp. 111-118, Singapore, 2013.

[91] H. Jadhav, U. Sharma, J. Patel, R. Roy, Brain storm optimization algorithm based economic dispatch considering wind power, In: Proceedings of the 2012 IEEE international conference on power and energy (PECon 2012), pp 588–593, Kota Kinabalu, Malaysia, 2012.

[92] K. Ramanand, K. Krishnanand, BK. Panigrahi, MK. Mallick, Brain storming incorporated teaching learning-based algorithm with application to electric power dispatch, In: BK. Panigrahi, S. Das, PN. Suganthan, PK. Nanda (eds), Swarm, evolutionary, and memetic computing, Vol. 7677, Lecture Notes in Computer Science, pp. 476–483, Springer, Berlin, Germany, 2012.

[93] AR. Jordehi, Brainstorm optimisation algorithm (BSOA): an efficient algorithm for finding optimal location and setting of facts devices in electric power systems, Electr Power Energy Syst, Vol. 69, pp. 48–57, 2015.

[94] GW. Zhang, ZH. Zhan, KJ. Du, WN. Chen, Normalization group brain storm optimization for power electronic circuit optimization, In: Proceedings of the 2014 conference companion on genetic and evolutionary computation companion, GECCO Comp '14ACM, pp. 183–184, New York, NY, USA, 2014.

[95] K. Lenin, BR. Reddy, MS. Kalavathi, Brain storm optimization algorithm for solving optimal reactive power dispatch problem, Int J Res Electron Commun Technol, Vol. 1, No. 3, pp. 25–30, 2014.

[96] X. Zhao, Research and application of brain storm optimization algorithm, Master's thesis, Xi'an University of Technology, 2013.

[97] M. Arsuaga-Ríos, MA. Vega-Rodríguez, Cost optimization based on brain storming for grid scheduling, Fourth edition of the International Conference on the Innovative Computing Technology (INTECH 2014), pp. 31-36, Luton, UK, 2014.

[98] Y. Sun, A hybrid approach by integrating brain storm optimization algorithm with grey neural network for stock index forecasting, Abstract Appl Anal, Vol. 2014, pp. 1–10, 2014.

[99] C. Sun, H. Duan, Y. Shi, Optimal satellite formation reconfiguration based on closed-loop brain storm optimization, IEEE Comput Intell Mag, Vol. 8, Issue 4, pp 39–51, 2013.

[100] J. Li, H. Duan, Simplified brain storm optimization approach to control parameter optimization in F/A-18 automatic carrier landing system, Aerosp Sci Technol, Vol. 42, pp. 187–195, 2015.

[101] H. Duan, C. Li, Quantum-behaved brain storm optimization approach to solving loney's solenoid problem, IEEE Trans Magn, Vol. 51, Issue 1, pp. 1–7, 2015.

[102] H. Duan, S. Li, Y. Shi, Predator-prey brain storm optimization for DC brushless motor, IEEE TransMagn, Vol. 49, Issue 10, pp. 5336–5340, 2013.

[103] K. M. Lynch, F. C. Park, Modern Robotics: Mechanics, Planning, and Control 1st Edition, Cambridge University Press, UK, 2017.

[104] K. Nonami, F. Kendoul, S. Suzuki, W. Wang, D. Nakazawa, Autonomous Flying Robots: Unmanned Aerial Vehicles and Micro Aerial Vehicles, Springer, 2010.

[105] C. Goerzen, Z. Kong, B. Mettler, A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance, J Intell Robot Syst, Vol. 57, pp. 65–100, 2010.

[106] K. Hwang Yong, A. Narendra, Gross Motion Planning- A Survey, ACM Computing Surveys, Vol. 24, No. 3, pp. 219-291, 1992.

[107] T. Lozano-Perez, M.A. Wesley, An algorithm for planning collision-free paths among polyhedral obstacles, Communications of the ACM, Vol. 22, Issue 10, pp. 560-570, 1979.

[108] V. Girbés, L. Armesto, J. Tornero, Path following hybrid control for vehicle stability applied to industrial forklifts, Robotics and Autonomous Systems, Vol. 62, Issue 6, pp. 910-922, 2014.

[109] X.Y. Wang, G.X. Zhang, J.B. Zhao, H.N. Rong, F. Ipate, R. Lefticaru, A Modified Membrane-Inspired Algorithm Based on Particle Swarm Optimization for Mobile Robot Path Planning, International Journal of Computers Communications and Control, Vol. 10, No. 5, pp. 732-745, 2015.

[110] XN. Bui, P. Souères, JD. Boissonnat, JP. Laumond, The Shortest path synthesis for non-holonomic robots moving forwards, [Research Report] RR-2153, INRIA, 1994.

[111] E.P. Anderson, R.W. Beard, T.W. McLain, Real-time dynamic trajectory smoothing for unmanned air vehicles, IEEE Transactions on Control Systems Technology, Vol. 13, Issue 3, pp. 471-477, 2005.

[112] M. Hwangbo, J. Kuffner, T. Kanade, Efficient Two-phase 3D Motion Planning for Small Fixed-wing UAVs, Proceedings 2007 IEEE International Conference on Robotics and Automation, pp. 1035-1041, Roma, Italy, 2007.

[113] M. Elbanhawi, M. Simic, R.N. Jazar, Continuous Path Smoothing for Car-Like Robots Using B-Spline Curves, Journal of Intelligent and Robotic Systems, Vol. 80, Supplement 1, pp. 23–56, 2015.

[114] J. Tang, J. Zhu, Z. Sun, A Novel Path Planning Approach Based on AppART and Particle Swarm Optimization, In: J. Wang, XF. Liao, Z. Yi (eds) Advances in Neural Networks – ISNN 2005, ISNN 2005, Lecture Notes in Computer Science, Vol. 3498, Springer, Berlin, Heidelberg, Germany, 2005.

[115] S. H. Tang, W. Khaksar, N. B. Ismail, M. K. A. Ariffin, A Review on Robot Motion Planning Approaches, Pertanika J. Sci. and Technol., Vol. 20, No. 1, pp. 15 – 29, 2012.

[116] L. Yang, J. Qi, D. Song, J. Xiao, J. Han, Y. Xia, Survey of Robot 3D Path Planning Algorithms, Journal of Control Science and Engineering, Vol. 2016, Article ID 7426913, 22 pages, 2016.

[117] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, J.P. How, Real-Time Motion Planning With Applications to Autonomous Urban Driving, IEEE Transactions on Control Systems Technology, Vol. 17, Issue 5, pp. 1105 - 1118 , 2009.

[118] M. K. Habib, H. Asama, Efficient method to generate collision free paths for an autonomous mobile robot based on new free space structuring approach, Intelligent Robots and Systems '91. 'Intelligence for Mechanical Systems, Proceedings IROS '91. IEEE/RSJ International Workshop on, pp. 563-567 Vol.2, Osaka, Japan, 1991.

[119] S. E. Muldoon, C. Luo, F. Shen, H. Mo, Naturally inspired optimization algorithms as applied to mobile robotic path planning, 2014 IEEE Symposium on Swarm Intelligence, pp. 1-6, Orlando, FL, USA, 2014.

[120] R.S. Tavares, T.C. Martins, M.S.G. Tsuzuki, Simulated annealing with adaptive neighborhood: A case study in off-line robot path planning, Expert Systems with Applications, Vol. 38, pp. 2951–2965, 2011.

[121] M. G. Park, M. C. Lee, Experimental evaluation of robot path planning by artificial potential field approach with simulated annealing, Proceedings of the 41st SICE Annual Conference, SICE 2002, pp. 2190-2195, Vol.4, Osaka, Japan, 2002.

[122] F. Janabi-Sharifi, D. Vinke, Integration of the artificial potential field approach with simulated annealing for robot path planning, Proceedings of 8th IEEE International Symposium on Intelligent Control, pp. 536-541, Chicago, IL, USA, 1993.

[123] C. Zheng, L. Li, F. Xu, F. Sun, M. Ding, Evolutionary Route Planner for Unmanned Air Vehicles, IEEE Transactions on Robotics, Vol. 21, No. 4, pp. 609-620, 2005.

[124] E. Besada-Portas, L. de la Torre, J. M. de la Cruz, B. de Andrés-Toro, Evolutionary Trajectory Planner for Multiple UAVs in Realistic Scenarios, IEEE Transactions on Robotics, Vol. 26, No. 4, pp. 619-634, 2010.

[125] A. Hosseinzadeh, H. Izadkhah, Evolutionary Approach for Mobile Robot Path Planning in Complex environment, International Journal of Computer Science Issues, Vol. 7, Issue 4, No. 8, pp. 1-9, 2010.

[126] M. Davoodi, F. Panahi, A. Mohades, S. N. Hashemi, Clear and smooth path planning, Applied Soft Computing, Vol. 32, pp. 568-579, 2015.

[127] Y. V. Pehlivanoglu, A new vibrational genetic algorithm enhanced with a Voronoi diagram for path planning of autonomous UAV, Aerospace Science and Technology, Vol. 16, pp. 47–55, 2012.

[128] Q. Li, W. Zhang, Y. Yin, Z. Wang, G. Liu, An Improved Genetic Algorithm of Optimum Path Planning for Mobile Robots, Sixth International Conference on Intelligent Systems Design and Applications, pp. 637-642, Jinan, China, 2006.

[129] A. Tuncer, M. Yildirim, Dynamic path planning of mobile robots with improved genetic algorithm, Computers and Electrical Engineering, Vol. 38, pp. 1564–1572, 2012.

[130] M. Chen, A. M. S. Zalzala, Safety considerations in the optimisation of paths for mobile robots using genetic algorithms, First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, pp. 299-306, Sheffield, UK, 1995.

[131] AL-Taharwa, A. Sheta, M. Al-Weshah, A Mobile Robot Path Planning Using Genetic Algorithm in Static Environment, Journal of Computer Science, Vol. 4, No. 4, pp. 341-344, 2008.

[132] F. Ahmed, K. Deb, Multi-objective optimal path planning using elitist non-dominated sorting genetic algorithms, Soft Computing, Vol. 17, Issue 7, pp. 1283–1299, 2013.

[133] Z. Cheng, Y. Sun, Y. Liu, Path planning based on immune genetic algorithm for UAV, 2011 International Conference on Electric Information and Control Engineering, pp. 590-593, Wuhan, China, 2011.

[134] K. Nikolos, K. P. Valavanis, N. C. Tsourveloudis, A. N. Kostaras, Evolutionary algorithm based offline/online path planner for UAV navigation, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), Vol. 33, No. 6, pp. 898-912, 2003.

[135] S. Mittal, K. Deb, Three-dimensional offline path planning for UAVs using multiobjective evolutionary algorithms, 2007 IEEE Congress on Evolutionary Computation, pp. 3195-3202, Singapore, 2007.

[136] A. Elshamli, H. A. Abdullah, S. Areibi, Genetic algorithm for dynamic path planning, Canadian Conference on Electrical and Computer Engineering 2004 (IEEE Cat. No.04CH37513), pp. 677-680, Vol.2, Ontario, Canada, 2004.

[137] Z. Yao, L. Ma, A Static Environment-Based Path Planning Method by Using Genetic Algorithm, 2010 International Conference on Computing, Control and Industrial Engineering, pp. 405-407, Wuhan, China, 2010.

[138] J. Cao, Y. Li, S. Zhao, X. Bi, Genetic-Algorithm-Based Global Path Planning for AUV, 2016 9th International Symposium on Computational Intelligence and Design (ISCID), pp. 79-82, Hangzhou, China, 2016.

[139] D. M. Pierre, N. Zakaria, A. J. Pal, Master-Slave Parallel Vector-Evaluated Genetic Algorithm for Unmanned Aerial Vehicle's path planning, 2011 11th International Conference on Hybrid Intelligent Systems (HIS), pp. 517-521, Melacca, Malaysia, 2011.

[140] V. Roberge, M. Tarbouchi, G. Labonte, Comparison of Parallel Genetic Algorithm and Particle Swarm Optimization for Real-Time UAV Path Planning, IEEE Transactions on Industrial Informatics, Vol. 9, No. 1, pp. 132-141, 2013.

[141] X. Zhang, H. Duan, An improved constrained differential evolution algorithm for unmanned aerial vehicle global route planning planning, Applied Soft Computing, Vol. 26, pp. 270-284, 2015.

[142] Z. Zhou, H. Duan, P. Li, B. Di, Chaotic differential evolution approach for 3D trajectory planning of unmanned aerial vehicle, 2013 10th IEEE International

Conference on Control and Automation (ICCA), pp. 368-372, Hangzhou, China, 2013.

[143] A. N. Brintaki, I.K. Nikolos, Coordinated UAV path planning using Differential Evolution, Operational Research, Vol. 5, Issue 3, pp. 487–502, 2005.

[144] J. Chakraborty, A. Konar, L. C. Jain, U. K. Chakraborty, Cooperative multi-robot path planning using differential evolution, Journal of Intelligent and Fuzzy Systems: Applications in Engineering and Technology - Theoretical advances of intelligent paradigms, Vol. 20, Issue 1,2, pp. 13-27, 2009.

[145] C. L. Huo, T. Y. Lai, T. Y. Sun, The preliminary study on multi-swarm sharing particle swarm optimization: Applied to UAV path planning problem, 2011 IEEE Congress of Evolutionary Computation (CEC), pp. 1770-1776, New Orleans, LA, USA, 2011.

[146] G. Han, W. Fu, W. Wang, The Study of Intelligent Vehicle Navigation Path Based on Behavior Coordination of Particle Swarm, Computational Intelligence and Neuroscience, Vol. 2016, Article ID 6540807, 10 pages, 2016.

[147] Y. Liu, X. Zhang, X. Guan, D. Delahaye, Adaptive sensitivity decision based path planning algorithm for unmanned aerial vehicle with improved particle swarm optimization, Aerospace Science and Technology, Vol. 58, pp. 92–102, 2016.

[148] Y. Zhang, D. Gong, J. Zhang, Robot path planning in uncertain environment using multi-objective particle swarm optimization, Neurocomputing, Vol. 103, pp. 172–185, 2013.

[149] Y. Fu, M. Ding, C. Zhou, Phase Angle-Encoded and Quantum-Behaved Particle Swarm Optimization Applied to Three-Dimensional Route Planning for UAV, IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, Vol. 42, No. 2, pp. 511-526, 2012.

[150] Y. Bao, X. Fu, X. Gao, Path planning for reconnaissance UAV based on Particle Swarm Optimization, 2010 Second International Conference on Computational Intelligence and Natural Computing, pp. 28-32, Wuhan, China, 2010.

[151] L. Guangshun, S. Hongbo, Path planning for mobile robot based on particle swarm optimization, 2008 Chinese Control and Decision Conference, Yantai, pp. 3290-3294, Shandong, China, 2008.

[152] M. Li, D. B. Wang, T. T. Bai, S. Z. Sheng, Route planning based on particle swarm optimization with threat heuristic, Electronics Optics and Control, Vol. 18, No. 12, pp. 1–4, 2011.

[153] C. Xin,Y.M. Li, Smooth path planning of a mobile robot using stochastic particle swarm optimization, Proceedings of the IEEE International Conference on Mechatronics and Automation, pp.1722– 1727, Luoyang, China, 2006.

[154] E. Masehian, D.Sedighizadeh, Multi-objective PSO and NPSO based algorithms for robot path planning, Adv.Electr.Comput.Eng., Vol. 10, pp. 69–76, 2010.

[155] D. Gong, J. Zhang, Y. Zhang, Multi-objective particles warm optimization for robot path planning in environment with danger sources, J. Comput., Vol. 6, No. 8, pp. 1554–1561, 2011.

[156] Q.R. Zhang, G.C. Gu, Path planning based on improved binary particle swarm optimization algorithm, Proceedings of IEEE International Conference on Robotics, Automation and Mechatronics, pp. 462–466, Chendu, China, 2008.

[157] Z. C. Fan, Path planning method based on the algorithm of PSO and elastic rope for underwater vehicle in three-dimensional space [Master thesis], Harbin Engineering University, China, 2013.

[158] M. Saska, M. Macas, L. Preucil, L. Lhotska, Robot Path Planning using Particle Swarm Optimization of Ferguson Splines, 2006 IEEE Conference on Emerging Technologies and Factory Automation, pp. 833-839, Prague, Czech Republic, 2006.

[159] N. Geng, D. W. Gong, Y. Zhang, PSO-Based Robot Path Planning for Multisurvivor Rescue in Limited Survival Time, Mathematical Problems in Engineering, Vol. 2014, Article ID 187370, 10 pages, 2014.

[160] L. Lu, D. Gong, Robot Path Planning in Unknown Environments Using Particle Swarm Optimization, 2008 Fourth International Conference on Natural Computation, pp. 422-426, Jinan, China, 2008.

[161] GZ. Tan, H. He, A. Sloman, Ant Colony System Algorithm for Real-Time Globally Optimal Path Planning of Mobile Robots, Acta Automatica Sinica, Vol. 33, Issue 3, pp. 279-285, 2007.

[162] H. Wang, W. Xiong, Research on global path planning based on ant colony optimization for AUV, Journal of Marine Science and Application, Vol. 8, No. 1, pp. 58–64, 2009.

[163] L. Yang, K.S. Li, W.S. Zhang, Y. Wang, Y. Chen, L.F. Zheng, Three-dimensional Path Planning for Underwater Vehicles Based on an Improved Ant Colony Optimization Algorithm, Journal of Engineering Science and Technology Review, Vol. 8, No. 5, pp.24-33, 2015.

[164] G. Zhang, H. Jia, Global path planning of AUV based on improved ant colony optimization algorithm, 2012 IEEE International Conference on Automation and Logistics, pp. 606-610, Zhengzhou, China, 2012.

[165] G. Zhang, H. Jia, 3D path planning of AUV based on improved ant colony optimization, Proceedings of the 32nd Chinese Control Conference, pp. 5017-5022, Xi'an, China, 2013.

[166] W. Ye, D. Ma, H. Fan, Algorithm for Low Altitude Penetration Aircraft Path Planning with Improved Ant Colony Algorithm, Chinese Journal of Aeronautics, Vol. 18, No. 4, pp. 304-309, 2005.

[167] H. Duan, X. Zhang, J. Wu, G. Ma, Max-Min Adaptive Ant Colony Optimization Approach to Multi-UAVs Coordinated Trajectory Replanning in Dynamic and Uncertain Environments, Journal of Bionic Engineering, Vol. 6, No. 2, pp. 161-173, 2009.

[168] M. Chen, Q. Wu, C. Jiang, A modified ant optimization algorithm for path planning of UCAV, Applied Soft Computing, Vol., No. 4, pp. 1712-1718, 2008.

[169] M.A. Porta Garcia, O. Montiel, O. Castillo, R. Sepúlveda, P. Melin, Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation, Applied Soft Computing, Vol. 9, No. 3, pp. 1102-1110, 2009.

[170] X. Fan, X. Luo, S. Yi, S. Yang, H. Zhang, Optimal path planning for mobile robots based on intensified ant colony optimization algorithm, IEEE International Conference on Robotics, Intelligent Systems and Signal Processing, 2003, Proceedings 2003, pp. 131-136, Vol.1, Changsha, Hunan, China, 2003.

[171] G. Wang, L. Guo, H. Duan L. Liu, H. Wang, A Modified Firefly Algorithm for UCAV Path Planning, International Journal of Hybrid Information Technology, Vol.5, No.3, pp. 123-144, 2012.

[172] C. Liu, Y. Zhao, F. Gao, L. Liu ,Three-Dimensional Path Planning Method for Autonomous Underwater Vehicle Based on Modified Firefly Algorithm, Mathematical Problems in Engineering, Vol. 2015, Article ID 561394, 10 pages, 2015.

[173] P.K. Mohanty, D.R. Parhi, Optimal path planning for a mobile robot using cuckoo search algorithm, Journal of Experimental and Theoretical Artificial Intelligence, Vol. 28, No. 1-2, pp. 35-52, 2016.

[174] B. Li, L. Gong, W. Yang, An Improved Artificial Bee Colony Algorithm Based on Balance-Evolution Strategy for Unmanned Combat Aerial Vehicle Path Planning, The Scientific World Journal, Vol. 2014, Article ID 232704, 10 pages, 2014.

[175] C. Xu, H Duan, F. Liu, Chaotic artificial bee colony approach to Uninhabited Combat Air Vehicle (UCAV) path planning, Aerospace Science and Technology, Vol. 14, pp. 535–541, 2010.

[176] B. Zhang, H. Duan, Three-Dimensional Path Planning for Uninhabited Combat Aerial Vehicle Based on Predator-Prey Pigeon-Inspired Optimization in Dynamic Environment, IEEE/ACM Transactions on Computational Biology and Bioinformatics, Vol. 14, No. 1, pp. 97-107, 2017.

[177] C. Purcaru, R.E. Precup, D. Iercan, L.O. Fedorovici, R.C. David, F.Drăgan, Optimal Robot Path Planning Using Gravitational Search Algorithm, International Journal of Artificial Intelligence, Vol. 10, No. S13, pp. 1-20, 2013.

[178] P. Li, H. Duan, Path planning of unmanned aerial vehicle based on improved gravitational search algorithm, Science China Technological Sciences, Vol. 55, No. 10, pp. 2712–2719, 2012.

[179] H. Duan, S. Liu, J. Wu, Novel intelligent water drops optimization approach to single UCAV smooth trajectory planning, Aerospace Science and Technology, Vol. 13, No. 8, pp. 442-449, 2009.

[180] Y. Zhang, G. Guan, X. Pu, The Robot Path Planning Based on Improved Artificial Fish Swarm Algorithm, Mathematical Problems in Engineering, Vol. 2016, Article ID 3297585, 11 pages, 2016.

[181] Y. Fu, M. Ding, C. Zhou, H. Hu, Route Planning for Unmanned Aerial Vehicle (UAV) on the Sea Using Hybrid Differential Evolution and Quantum-Behaved Particle Swarm Optimization, IEEE Transactions on Systems, Man, and Cybernetics: Systems, Vol. 43, No. 6, pp. 1451-1465, 2013.

[182] B. Tang, Z. Zhu, J. Luo, Hybridizing Particle Swarm Optimization and Differential Evolution for the Mobile Robot Global Path Planning, International Journal of Advanced Robotic Systems, Vol. 13, No. 3, pp. 1-17, 2016.

[183] H. Mo, L. Xu, Research of biogeography particle swarm optimization for robot path planning, Neurocomputing, Vol. 148, pp. 91–99, 2015.

[184] P.K.Das, H.S.Behera, S. Das, H.K.Tripathy, B.K.Panigrahi, S.K.Pradhan, A hybrid improved PSO-DV algorithm for multi-robot path planning in a clutter environment, Neurocomputing, Vol. 207, pp. 735-753, 2016.

[185] G. Wang, L. Guo, H. Duan, L. Liu, H. Wang, A Bat Algorithm with Mutation for UCAV Path Planning, The ScientificWorld Journal, Vol. 2012, Article ID 418946, 15 pages, 2012.

[186] S. Piao, B. Hong, Path planning of robot using genetic annealing algorithm, Proceedings of the 4th World Congress on Intelligent Control and Automation (Cat. No.02EX527), pp. 493-495, Vol.1, Shanghai, China, 2002.

[187] H. Duan, Y. Yu, X. Zhang, and S. Shao, Three-dimension path planning for UCAV using hybrid meta-heuristic ACODE algorithm, Simulation Modelling Practice and Theory, Vol. 18, No. 8, pp. 1104–1115, 2010.

[188] B.K. Oleiwi, H.Roth, B.I.Kazem, A hybrid approach based on ACO and GA for multi-objective mobile robot path planning, Appl.Mech.Mater., Vol. 527, pp. 203–212, 2014.

[189] C. Purcaru, R. E. Precup, D. Iercan, L. O. Fedorovici, R. C. David, Hybrid PSO-GSA robot path planning algorithm in static environments with danger zones, 2013 17th International Conference on System Theory, Control and Computing (ICSTCC), pp. 434-439, Sinaiapp, Romania, 2013.

[190] M. Ju, S. Wang, J. Guo, Path Planning Using a Hybrid Evolutionary Algorithm Based on Tree Structure Encoding, The Scientific World Journal, Vol. 2014, Article ID 746260, 8 pages, 2014.

[191] G. Wang, L. Guo, H. Duan, H. Wang, L. Liu, and M. Shao, A hybrid meta-heuristic DE/CS algorithm for UCAV three dimension path planning, The Scientific World Journal, Vol.2012, Article ID 583973, 11 pages, 2012.

[192] H. Qiu, H. Duan Receding horizon control for multiple UAV formation flight based on modified brain storm optimization, Nonlinear Dyn, Vol. 78, pp. 1973–1988, 2014.

[193] H. Qiu, H. Duan, Y. Shi, A decoupling receding horizon search approach to agent routing and optical sensor tasking based on brain storm optimization, Optik, Vol. 126 pp. 690–696, 2015.

[194] H. Duan, L. Huang, Imperialist competitive algorithm optimized artificial neural networks for UCAV global path planning, Neurocomputing, Vol 125, pp. 166-171, 2014.

[195] A. Alihodzic, E. Tuba, R. Capor-Hrosik, E. Dolicanin, M. Tuba, Unmanned aerial vehicle path planning problem by adjusted elephant herding optimization, in Proceedings of the 25th Telecommunications Forum (TELFOR), pp. 804–807, 2017.

[196] M. A. Contreras-Cruz, V. Ayala-Ramirez, U. H. Hernandez-Belmonte, Mobile robot path planning using artificial bee colony and evolutionary programming, Applied Soft Computing, vol. 30, pp. 319–328, 2015.

[197] Z. Nie, X. Yang, S. Gao, Y. Zheng, J. Wang, and Z. Wang, Research on autonomous moving robot path planning based on improved particle swarm optimization, in IEEE Congress on Evolutionary Computation (CEC), pp. 2532–2536, IEEE, 2016.

Biografija

Irfan Fetahović je rođen 13.06.1982. godine u Novom Pazaru, Republika Srbija. Osnovnu školu i Gimnaziju završio u Novom Pazaru. Diplomirao na Elektrotehničkom fakultetu Univerziteta u Beogradu, Odsek za računarsku tehniku i informatiku 2007. godine. Od 2007. god. radi na Državnom univerzitetu u Novom Pazaru, prvo kao asistent pripravnik, zatim u zvanju saradnika u nastavi, a od 2010. do 2016. god. je bio asistent za užu naučnu oblast računarska tehnika na Departmanu tehničkih nauka. Doktorske studije upisao 2015. godine na DUNP, na studijskom programu računarska tehnika.