Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko

Doktorska disertacija

Diferencialna evolucija za rekonstrukcijo parametriziranih proceduralnih drevesnih modelov

Maj 2012

mag. Aleš Zamuda

Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko

Doktorska disertacija

Diferencialna evolucija za rekonstrukcijo parametriziranih proceduralnih drevesnih modelov

Avtor mag. Aleš Zamuda
Mentor red. prof. dr. Janez Brest

Maj 2012
UDK 004.8.021:004.925.8(043.3)

Avtor:	mag. Aleš Zamuda, univ. dipl. inž. rač. in inf.
Naslov:	Diferencialna evolucija za rekonstrukcijo parametriziranih proceduralnih drevesnih modelov
UDK:	004.8.021:004.925.8(043.3)
Ključne besede:	Diferencialna evolucija, drevo, proceduralni model, rekonstrukcija strukture, numerično kodiranje, večkriterijska optimizacija, evolucijski algoritmi
Število izvodov:	9

Univerza v Mariboru

Maribor, 20. 12. 2011
Številka: DR 111/2011/425-MGM

Na osnovi 287., 140., 142. in 144. člena Statuta Univerze v Mariboru (Statut UM-UPB8, Ur. l. RS, št. 1/2010) ter sklepa 6. redne seje Senata Univerze v Mariboru z dne 20. 12. 2011 v zvezi z vlogo doktorskega kandidata mag. Aleša Zamude za sprejem odločitve o predlagani temi doktorske disertacije in mentorja,

izdajam naslednji

SKLEP

Odobri se tema doktorske disertacije mag. Aleša Zamude s Fakultete za elektrotehniko, računalništvo in informatiko z naslovom »Diferencialna evolucija za rekonstrukcijo parametriziranih proceduralnih drevesnih modelov«. Za mentorja se imenuje red. prof. dr. Janez Brest. Kandidat mora članici predložiti izdelano doktorsko disertacijo v zadostnih izvodih najpozneje do 19. 12. 2015.

Obrazložitev:

Kandidat mag. Aleš Zamuda je dne 4. 7. 2011 na Fakulteti za elektrotehniko, računalništvo in informatiko vložil vlogo za potrditev teme doktorske disertacije z naslovom »Diferencialna evolucija za rekonstrukcijo parametriziranih proceduralnih drevesnih modelov«. Za mentorja je bil predlagan red. prof. dr. Janez Brest.

Senat Fakultete za elektrotehniko, računalništvo in informatiko je na osnovi pozitivnega mnenja komisije za oceno teme doktorske disertacije, ki je ugotovila, da kandidat izpolnjuje pogoje za pridobitev doktorata znanosti, in ocenila, da je predlagana tema ustrezna, sprejel pozitivno mnenje in poslal predlog teme doktorske disertacije s predlogom mentorja v odobritev Senatu univerze.

Senat Univerze v Mariboru je po proučitvi vloge in na osnovi določil Statuta Univerze v Mariboru sprejel svojo odločitev o predlagani temi doktorske disertacije in imenoval mentorja, kot izhaja iz izreka.

V skladu s 144. členom Statuta Univerze v Mariboru mora kandidat za pridobitev doktorata znanosti najpozneje v štirih letih od dneva izdaje tega sklepa, članici predložiti izdelano doktorsko disertacijo v zadostnih izvodih. Kandidatu je bil določen rok glede na datum sprejetja teme na pristojnem organu.

Pouk o pravnem sredstvu:
Zoper ta sklep je možna pritožba na Senat Univerze v Mariboru v roku 8 dni od prejema tega sklepa.

Rektor:
Prof. dr. Danijel Rebolj

Obvestiti:
1. Kandidata.
2. Fakulteto.
3. Arhiv.

UNIVERZA V MARIBORU 17

HR EXCELLENCE IN RESEARCH

Zahvala

Najprej se zahvaljujem materi in očetu, ki sta me vzgojila v posameznika, kakršen sem se trudil postati. Hvala za vsa odrekanja, vzpodbude in čudovito otroštvo. Velika zahvala očetu, ki me je navdušil za računalništvo in mi pokazal programiranje.

Zahvaljujem se mentorju te razprave, red. prof. dr. Janezu Brestu, za vsako odkrito pot in strokovno pomoč v času njene izdelave. Zahvala prav tako sodelavcem v laboratoriju za računalniške arhitekture in jezike. Zahvalil bi se tudi številnim udeležencem konferenc na teme iz obravnavanih področij, s katerimi sem imel priložnost govoriti in so me dodatno spodbudili k delu. Zahvaljujem se vključenim, ki me niso ovirali pri raziskovanju in razmišljanju, in še mnogo večja zahvala tistim, ki so mi pri prizadevanjih tudi pomagali.

Še posebej se zahvaljujem vsem velikanom, na katerih ramena sem imel priložnost stopiti, da so mi pomagali pri vseh izobraževanjih. Zahvaljujem se učiteljem mentorjem iz časa javnega osnovnega šolanja in zasebnega dodatnega šolanja, kjer sem se naučil tudi precej temeljnih znanj iz računalništva; še posebej očetu, prvemu in večnemu mentorju; ter organizatorjem poletnih šol. Zahvala tudi podjetjem, pri katerih sem se imel možnost izpopolnjevati, t.j. HERMES SoftLab in IxtlanTeam d.o.o. Zahvala vsem predavateljem in asistentom na študijskih programih, ki sem jih obiskal na FERI, mnogi izmed njih so bili izjemni motivatorji. Ker disertacija gradi na mojem diplomskem in magistrskem delu in so tukaj vključena nekatera slikovna in druga gradiva iz obeh nalog, se zahvaljujem še mentorjema in vsem drugim, ki so mi pomagali pri obeh delih.

Zahvala članom komisije za oceno teme te doktorske disertacije in oceno predložene disertacije. Zahvala še avtorjem uporabljenih prosto dostopnih programskih kod algoritmov: red. prof. dr. Janezu Brestu za program jDE, mag. Tei Tušar za program DEMO, Ponnuthura Nagaratnam Suganthanu za priprave ogrodij za tekmovanja CEC in razpoložljivost ob delu na njih ter vsej odprtokodni skupnosti za odprtost in dostopnost GNU/Linux orodij.

Zahvala tudi vsem avtorjem dostopnih avtorskih del, s katerimi sem se imel priložnost seznaniti in so navedena kot uporabljena literatura. Zahvala prav tako vsem prijateljem in do sedaj neimenovanim, ki so me kakorkoli spodbujali in mi pomagali doseči, kar je dobro.

Naslov: Diferencialna evolucija za rekonstrukcijo parametriziranih proceduralnih drevesnih modelov

Ključne besede: Diferencialna evolucija, drevo, proceduralni model, rekonstrukcija strukture, numerično kodiranje, večkriterijska optimizacija, evolucijski algoritmi

Povzetek

V tej disertaciji obravnavamo tezo o lastnem razvoju novega evolucijskega algoritma, ki omogoča modeliranje z rekonstrukcijo parametriziranih proceduralnih modelov olesenelih rastlin iz slikovnih projekcij referenčnih modelov. Pokažemo, da je s predlaganim proceduralnim modelom po kodiranju v genotip možno izvesti evolucijski proces iskanja parametriziranih proceduralnih modelov. S postavitvijo ustreznostne funkcije in preslikavo genotipa v fenotip pokažemo, da je algoritem diferencialne evolucije primeren za iterativno rekonstrukcijo izbranega parametriziranega proceduralnega modela. Pokažemu tudi, da je algoritem diferencialne evolucije še posebej primeren za evolucijo izbranega modela, saj je ta fiksne dimenzije in parametre kodiramo v domeno realnih števil, za katero je algoritem diferencialne evolucije posebej učinkovito načrtovan. Potrdimo še, da je algoritem jDE s samoprilagodljivimi krmilnimi parametri za naš optimizacijski postopek ustreznejši od osnovnega algoritma DE brez samoprilagajanja krmilnih parametrov. Dobljeni rezultati potrjujejo, da je predstavljen pristop primeren za modeliranje drevesnih rastlin za računalniško animacijo, s pomočjo evolucije numerično kodiranega proceduralnega modela. S tem ugotovimo, da teza disertacije pritrdilno utemeljuje zastavljene hipoteze. Izsledke, ki smo jih pokazali v tej disertaciji, smo objavili tudi širši znanstveni javnosti.

Title: Differential Evolution for Parameterized Procedural Woody Plant Models Reconstruction

Keywords: Differential evolution, woody plant, tree, procedural model, structure reconstruction, numerical encoding, multi-objective optimization, evolutionary algorithms

Abstract

In this dissertation, we convey thesis on our own development of a new evolutionary algorithm, which enables modelling of woody plants using parameterized procedural models reconstruction from image projections of reference models. We show, that after encoding the proposed procedural model in a genotype, it is possible to create an evolutionary process to search for parameterized procedural models. By composing a fitness function and genotype-phenotype mapping, we show that differential evolution (DE) algorithm is suitable for iterative reconstruction of the chosen parameterizable procedural model. We also show that the DE algorithm is especially useful for evolution of the chosen model, since it is of fixed dimension and encodes the parameters in real number domain, for which DE algorithm is efficiently designed. We also confirm, that jDE algorithm with self-adaptive control parameters is better than basic DE algorithm without self-adapting control parameters. Obtained results confirm, that the proposed approach is suitable for modeling of woody plants for computer animation, using evolution of a numerically coded procedural model. By this we confirm, that the thesis of this dissertation confirms the set hypotheses. We have also published our findings displayed in this dissertation to a wider scientific public.

Kazalo

Slike

Tabele

Algoritmi

Kratice

ABC	ang. *Artificial Bee Colony* (optimizacija s kolonijami čebel)
AC	lat. *Ante Cristo* (pred našim štetjem)
ACO	ang. *Particle Swarm Optimization* (optimizacija s kolonijo mravelj)
AD	lat. *Anno Domini* (našega štetja)
AI	ang. *Artificial Intelligence* (umetna inteligenca)
AIS	ang. *Artificial Immune Systems* (umetni imunski sistemi)
ASOC	ang. *Applied Soft Computing*
AWD	ang. *Artificial Weed Colonies* (umetne travne kolonije)
BFO	ang. *Bacterial Foraging Optimization* (bakterijsko preiskovanje)
CEC	ang. *Congress on Evolutionary Computation*
CI	ang. *Computational Intelligence* (računska inteligenca)
CS	ang. *Cuckoo Search* (kukavičje iskanje)
CUDA	ang. Compute Unified Device Architecture (enotna računska strojna arhitektura)
DASA	ang. *Differential Ant-Stigmergy Algorithm* (algoritem diferencialne stigmergije mravelj)
DE	ang. *Differential Evolution* (diferencialna evolucija)
DECMOSA-SQP	ang. *Differential Evolution with Self-adaptation and Local Search for Constrained Multiobjective Optimization* (diferencialna evolucija s samoprilagajanjem in lokalnim preiskovanjem za večkriterijsko optimizacijo z omejitvami)
DEMO	ang. *Differential Evolution for Multi-objective Optimization* (diferencialna evolucija za večkriterijsko optimizacijo)
DEMOwSA	ang. *Differential Evolution for Multi-objective Optimization with Self-adaptation* (diferencialna evolucija za večkriterijsko optimizacijo s samoprilagajanjem)

DEwSAcc	ang. *Differential Evolution with Self-adaptation and Cooperative Coevolution* (diferencialna evolucija s samoprilagajanjem krmilnih parametrov in koevolucijo)
EC	ang. *Evolutionary Computation* (evolucijsko računanje)
EP	ang. *Evolutionary Programming* (evolucijsko programiranje)
ES	ang. *Evolution Strategies* (evolucijske strategije)
FA	ang. *Firefly Algorithm* (algoritem kresničke)
FEs	ang. *Fitness Evaluations* (ovrednotenja ustreznosti)
FERI	Fakulteta za elektrotehniko, računalništvo in informatiko (ang. Faculty of Electrical Engineering and Computer Science)
GA	ang. *Genetic Algorithm* (genetski algoritem)
GCC	ang. *GNU Compiler Collection* (zbirka prevajalnikov GNU)
GDE	ang. *Generalized Differential Evolution* (posplošena diferencialna evolucija)
GECCO	ang. *Genetic and Evolutionary Computation Conference*
GIS	ang. *Geographic Information System* (geografski informacijski sistem)
GP	ang. *Genetic Programming* (genetsko programiranje)
GO	ang. *Global Optimization* (globalna optimizacija)
HS	ang. *Harmony Search* (harmonično iskanje)
GPL	GNU General Public License
IEEE	ang. *Institute of Electrical and Electronics Engineers* (Inštitut inženirjev elektrotehnike in elektronike)
IS	ang. *Information Sciences*
jDE	ang. *self-adaptive control parameters differential evolution* (diferencialna evolucija s samoprilagodljivimi krmilnimi parametri); Janez DE
$\text{jDE}_{\text{NP,MM}}$	algoritem DE z zmanjševanjem velikosti populacije in več mutacijskih strategij
LiDAR	ang. *Light Detection And Ranging* (lasersko skeniranje površja)
LEM	ang. *Learnable Evolution Model* (učljiv evolucijski model)
LOD	ang. Level of Detail (stopnja podrobnosti)
MA	ang. *Memetic Algorithm* (memetski algoritem)
MO	ang. *Multi-objective Optimization* (večkriterijska optimizacija)

MOPSO	ang. *Multiobjective Particle Swarm Optimization* (večkriterijska optimizacija z rojem delcev)
MOEA/D	ang. *Multiobjective Evolutionary Algorithm based on Decomposition* (večkriterijski evolucijski algoritem bazeč na dekompoziciji)
MOjDE	ang. *Multi-objective Self-adaptive Differential Evolution* (samoprilagodljiva diferencialna evolucija za večkriterijsko optimizacijo)
MP	ang. *Mathematical Programming* (matematično programiranje)
MSVC	ang. *Microsoft Visual C++*
MTS	ang. Multiple Trajectory Search (večtrajektorijsko iskanje)
NSGA-II	ang. *Elitist Non-Dominated Sorting Genetic Algorithm* (elitističen nedominantno urejajoči genetski algoritem)
NURBS	ang. *Non-uniform rational B-spline* (neenakomerna racionalna osnovna krivulja)
PDE	ang. *Pareto-frontier Differential Evolution* (Pareto frontalna diferencialna evolucija)
PSO	ang. *Particle Swarm Optimization* (optimizacija z rojem delcev)
R8SPC	ang. *Region 8 Student Paper Contest* (tekmovanje študenstkih člankov regije 8 – IEEE)
SA-DE	ang. self-adaptive differential evolution (diferencialna evolucija z logaritmično normalnim samoprilagajanjem krmilnih parametrov)
SC	ang. *Soft Computing* (mehko računanje)
SI	ang. *Swarm Intelligence* (rojna inteligenca)
SPEA2	ang. *Strength Pareto Evolutionary Algorithm* (močni Pareto evolucijski algoritem)
SQP	ang. *Sequential Quadratic Programming* (sekvenčno kvadratično programiranje)
SWEVO	ang. *Swarm and Evolutionary Computation*
TEVC	ang. *Transactions on Evolutionary Computation*
UM	Univerza v Mariboru (ang. *University of Maribor*)
WCCI	ang. *World Congress on Computational Intelligence*

Oznake in simboli

cc_j	določilo o vključitvi j-te kriterijske komponente v koevolucijo
CR	krmilni parameter križanja
$\langle CR_g \rangle_i$	povprečna vrednost parametra CR v naboru vektorjev za križanje
D	dimenzija problema, ki ga rešujemo
$d_{i,g}$	dekompozicijska stopnja vektorja i v generaciji g
F	krmilni parameter mutacije
f	funkcija oz. problem, ki ga rešujemo
$f_{\min,i}$	najmanjša vrednost funkcijske komponente i
$f_{\max,i}$	največja vrednost funkcijske komponente i
$f'(x)$	odvod funkcije s parametrom x (po parametru x)
$f^*(x)$	optimiran približek optimuma funkcije $f(x)$
$\mathbf{f}(\mathbf{x})$	večkriterijska funkcija z vektorskim parametrom $\mathbf{x}$
FEs	največje število ovrednotenj v evolucijskem procesu
F_l	spodnja mejna vrednost parametra F
F_u	dolžina intervala za vrednost parametra F
$\langle F_g \rangle_i$	povprečna vrednost parametra F v naboru vektorjev za mutacijo
g	trenutna generacija evolucijskega procesa
G	skupno število generacij za evolucijski proces
G_c	številka generacije, do katere dovolimo rahljanje dopustnosti
$G_{i,g}^{\mathrm{cc}}$	število generacij, dokler je dekompozicija za koevolucijo fiksirana
G_LS	število generacij, na koliko izvajamo lokalno preiskovanje
$\mathbf{g}(\mathbf{x})$	funkcija za definicijo omejitev nad $\mathbf{x}$
$G_i(\mathbf{x})$	kršitev i-te omejitve, tudi neenakostne
$H_i(\mathbf{x})$	kršitev i-te enakostne omejitve

$g_i(\mathbf{x})$	i-ta omejitev, tudi neenakostna
$h_i(\mathbf{x})$	i-ta enakostna omejitev
j_{rand}	indeks komponente, ki je pri križanju ne izmenjamo
L	dimenzija vektorjev s pridruženimi krmilnimi parametri
l	dolžina zaporedja pri eksponentem križanju
M	število optimiranih kriterijev
$miter_i$	število iteracij lokalnega preiskovanja
NP	velikost populacije
$\Re^M$	prostor kriterijev dimenzije M
$\mathbf{p}$	posameznik iz populacije
$pmax$	število zmanjševanj populacije
$rand_j$	j-ti različen naključni indeks posameznika iz populacije
r_s	naključno število za izbiro mutacijske strategije
s	strategija algoritma DE
$U(a, b)$	uniformna naključna porazdelitev na intervalu $[a, b]$
$\mathbf{u}$	vektor mutacije
$u_{i,j,g+1}$	j-ta komponenta i-tega vektorja križanja za generacijo $g + 1$
$\mathbf{v}$	vektor križanja
$\overline{v}$	povprečna vrednost kršenja vseh omejitev
w_i	utež i-te kriterijske komponente za prednostni pristop
$\mathbf{x}_{\text{best}}$	vektor z najboljšo oceno
$\mathbf{x}_i$	i-ti posameznik trenutne populacije
$x_{i,j}$	j-ti elementi vektorja i-tega posameznika v trenutni populaciji
$\mathbf{x}_{\text{max}}$	vektor zgornjih mej intervalov
$\mathbf{x}_{\text{min}}$	vektor spodnjih mej intervalov
α	hitrost prilagajanje meje dopustnosti
β	število najboljših vektorjev za prilagajanje meje dopustnosti
ϵ_i	dopustni interval kršitve i-te omejitve
τ	stopnja prilagajanja
τ_1, τ_2	mejni verjetnosti zamenjave krmilnih parametrov F in CR
θ	začetno število najboljših vektorjev za določitev meje dopustnosti

$\mathbf{b}$	smer in velikost glavne veje
d	premer veje
g	Graveliusov red (indeks) veje
$\overline{\mathbf{g}}$, $\overline{\mathbf{h}}$, $\overline{\mathbf{k}}$,	enotski vektorji v smeri gravimorfizma, fototropizma, ortotropizma,
$\overline{\mathbf{p}}$, $\overline{\mathbf{w}}$	plagiotropizma in popačenja
$k_{\mathrm{l}}^{g,w}$	skalirni faktor dolžine veje
$k_{\mathrm{c}}^{g,w}$	koeficient gravicentralizma za veje na indeksu (g, w)
k_{p}	fototropizem (tendenca rasti v smeri proti svetlobi)
k_{f}	prožnost vej drevesa pri vetru
$k_{\mathrm{o}}^{g,w}$	ortotropizem (t. j. tendenca po rasti navpično navzgor)
$k_{\mathrm{p}}^{g,w}$	plagiotropizem (t. j. tendenca po rasti horizontalno navzven od debla)
$k_{\mathrm{s}}^{g,w}$	delež žil osnovne veje, ki se cepijo v glavno podvejo
k_{d}	koeficient debeline veje
k_{w}	Writhov koeficient za naključnost rasti
L_0	relativna dolžina osnovne veje
L_1 (L_2)	relativna dolžina glavne (stranske) podveje
$l_0^{0,0}$	dolžina prvega odseka debla
l_0	dejanska dolžina osnovne veje
l_1 (l_2)	dejanska dolžina glavne (stranske) podveje
l_{type}	način porazdelitve listov
l_{l}	velikost listov
l_{LOD}	stopnja podrobnosti
t_{bark} in t_{leaf}	identifikatorja teksture debla in listov
$\mathbf{M}_{\mathrm{m};1}^{-1}$	inverzna matrika za konstruiranje vektorja proti tlom v koordinatnem sistemu glavne podveje
$\mathbf{M}_{\mathbf{w};1}^{-1}$	inverzna matrika za konstrukcijo vektorja pihanja vetra v koordinatnem sistemu glavne podveje
M_0	koordinatni sistem prvega odseka debla (globalni koordinatni sistem)
$m_t^{g,w}$ ($M_t^{g,w}$)	spodnja (zgornja) meja relativne dolžine podvej glede na dolžino osnovne veje, ločeno za tip podveje ($t \in \{major, minor\}$)
n	število strani piramide za upodobitev prisekanih stožcev vej drevesa

$n_t^{g,w}$ ($N_t^{g,w}$)	spodnja (zgornja) meja absolutne dolžine podvej glede na dolžino osnovne veje, ločeno za tip podveje ($t \in \{major, minor\}$)
$m^{g,w}$ ($M^{g,w}$)	spodnja (zgornja) meja relativne dolžine podvej glede na dolžino osnovne veje združeno za tip podveje
$\mathbf{q}$	smer veje
r_1 (r_2)	relativna dolžina glavne (stranske) podveje do naslednje vejitve
S	število žil v deblu drevesa (začetni odsek do prve vejitve)
S_0	število žil v trenutni veji, ki je lahko tudi deblo
S_1	število žil, ki se odcepi od trenutne veje v glavno podvejo
S_2	število žil, ki se odcepi od trenutne veje v stransko podvejo
w	Weibullov red (indeks) veje
$\mathbf{w}_0$	smer pihanja vetra, v koordinatnem sistemu trenutne veje
$w_\mathrm{s}(t)$	hitrost (jakost) vetra spremenljive smeri na drevo
$w_\mathrm{g}(t)$	hitrost (jakost) pihanja sunka vetra nespremenljive smeri na drevo
w_x (w_y)	kot zasuka trenutne veje vzdolž osi veje (pravokotno na vejo) zaradi neusmerjenega pihanja vetra
w_geom	Weibullov indeks, do katerega razvijamo geometrijsko strukturo
w_branch	Weibullov indeks, do katerega rišemo veje drevesa
$\alpha^{g,w}$	kot med izhajajočima podvejama za veje na indeksu (g, w)
$\alpha_\mathrm{m}^{g,w}$	kot zasuka za gravimorfizem za veje na indeksu (g, w)
α_1	kot odklona glavne podveje
α_2	kot odklona stranske podveje
α_p	kot zasuka okoli osi vzdolž veje
$\alpha_x(t)$, $\alpha_z(t)$	kota zasuka pri vetru spremenljive smeri
α_w	kot zasuka pri usmerjenem vetru
ρ_l	gostota listov

1. Uvod

Najtežje je biti preprost. (Eyecatching)

V tej znansteni razpravi za dosego doktorskega naslova (disertaciji) osvetljujemo problematiko avtomatskega modeliranja naravnih objektov na računalniku. Omejimo se na rekonstrukcijsko modeliranje trodimenzionalnih modelov olesenelih rastlin (dreves). Do nedavnega je bilo za modeliranje dreves na računalniku potrebno podati veliko število parametrov modela, bodisi numerično, bodisi grafično. Ob vse večji računalniški moči in združevanju algoritmičnih vzorcev v abstraktne sisteme se postavi vprašanje, kako avtomatizirati parametrizacijo parametrov modela glede na obstoječ slikovni material. Temu se ponuja odgovor v luči modeliranja z rekonstrukcijo proceduralnih modelov iz slik. Kot vhod uporabnik poda dvodimenzionalni slikovni material in modelirni postopek samodejno izračuna parametre trodimenzionalnega modela kot izhod. Pri tem modeliranje trodimenzionalne strukture drevesa zahteva ne le rekonstrukcijo oblaka točk v prostoru, temveč natančnejšo predstavitev topologije in drugih lastnosti modela, ki so odvisne od oblikovnih značilnosti rekonstruiranega objekta. V obravnavani temi povezujemo evolucijske algoritme z računalniško grafiko in animacijo skozi računalniški vid, obdelavo slik in fotogrametrijo ter si pri tem pomagamo s tehnikami iz robotike in navidezne resničnosti, kar ob pohodu računske inteligence in vseprisotnega računalništva napoveduje široko uporabnost izbrane teme.

Predlagamo nov pristop k modeliranju strukture geometrijskih modelov olesenelih rastlin s pomočjo rekonstrukcije parametriziranih proceduralnih modelov. Rekonstrukcija iz dvodimenzionalnih slik (projekcij) z iterativnim optimizacijskim postopkom tvori trodimenzionalni geometrijski model. Kot optimizacijski postopek za parametrizacijo proceduralnega modela smo izbrali algoritem diferencialne evolucije. S tem algoritmom ovrednotimo parametriziran proceduralni drevesni model. To storimo tako, da poiščemo ustreznen nabor parametrov, s katerimi zgradimo čim bolj podobno drevo referenčnemu drevesu. Uporabljen proceduralni model s proceduralnim postopkom rekurzivno izračuna vse sestavne dele trodimenzionalne drevesne strukture. Ponavlja našo proceduro za drevesni model nad dano veliko množico numerično kodiranih vhodnih parametrov, ki služijo kot seme. Parametriziran proceduralni model je nato možno animirati. Primerjavo modela in referenčnega drevesa opravimo na nivoju slikovnih

elementov (pikslov) projekcij, tako da seštejemo razdalje med najbližjimi podobnimi piksli. V disertaciji tako predstavimo nov rekonstrukcijski postopek in po opravljenih raziskavah pridemo do naslednjih izvirnih znanstvenih prispevkov, ki so empirično pokazljivi:

- priprava lastnega modela in upodabljalnika dreves, ki omogoča hitro animacijo in s tem vzorčenje razvijanih primerkov,
- kodiranje parametrov našega proceduralnega modela v evolucijskem algoritmu,
- razvoj algoritma za izris projekcij parametriziranega proceduralnega drevesa,
- algoritem za primerjavo optimiranih modelov trodimenzionalnih dreves z referenčnimi projekcijami,
- uporaba samoprilagodljive diferencialne evolucije v rekonstrukcijem postopku in
- primerjava kakovosti diferencialne evolucije s samoprilagodljivimi krmilnimi parametri napram nespremenljivim krmilnim parametrom za naš optimizacijski postopek.

Pri rekonstrukciji naravnih objektov se omejimo na rekonstrukcijo drevesnih modelov. Omejimo se na postopke, ki so trenutno izračunljivostno učinkoviti, zato ne uporabljamo genetskega programiranja in semen proceduralnih modelov s spremenljivo dolžino. Pri upodabljanju dreves se omejimo na uporabo odprtokodnega paketa EcoMod in pri evolucijskih algoritmih na diferencialno evolucijo. Pri rekonstrukciji predpostavljamo določeno mero samopodobnosti sestavnih delov dreves, ki jo kodiramo v rekurzivni proceduri razvitega modela.

Potrebne raziskave izvedemo tako, da predstavljen rekonstrukcijski algoritem implementiramo in preizkusimo s pomočjo odprtokodnega paketa EcoMod. Vanj vključimo algoritem diferencialne evolucije in poskrbimo za pretvorbo podatkov med algoritmom diferencialne evolucije in algoritmom za upodabljanje drevesnega modela. Prosto dostopen odprtokodni programski paket EcoMod je dostopen na spletnem naslovu `http://ecomod.sourceforge.net`. Dobljene rezultate raziskav smo prav tako že predstavili strokovni javnosti, s prispevki na nacionalnih in mednarodnih konferencah in v mednarodnih znanstvenih revijah z visokim faktorjem vpliva v znanstveni kategoriji računalništvo, podkategorij umetna inteligenca, interdisciplinarne aplikacije

ter teorija in metode. Objave segajo tudi v kategorijo avtomatizacije in krmiljenja sistemov ter operacijskih raziskav in upravljanja.

Temeljni cilj in zanj obravnavana teza doktorske disertacije je tako razviti evolucijski algoritem, ki bo omogočal modeliranje z rekonstrukcijo parametriziranih proceduralnih modelov olesenelih rastlin iz slikovnih projekcij referenčnih objektov. Za potrditev teze doktorske disertacije zato preskusimo veljavnosti naslednjih hipotez:

> **Hipoteza 1**: z našim proceduralnim modelom je možno izvesti evolucijski proces iskanja parametriziranih proceduralnih modelov.
>
> **Hipoteza 2**: algoritem diferencialne evolucije je primeren za iterativno rekonstrukcijo parametriziranih proceduralnih modelov dreves.
>
> **Hipoteza 3**: algoritem diferencialne evolucije s samoprilagodljivimi krmilnimi parametri je za naš postopek učinkovitejši od osnovnega algoritma diferencialne evolucije brez samoprilagajanja krmilnih parametrov.

V nadaljevanju podajamo izčrpen pregled sorodnih del in ozadja obravnavanih raziskovalnih področjih. V tretjem poglavju predstavimo predlagan nov pristop za reševanje zastavljenega problema. V četrtem poglavju predstavimo rezultate in ocenimo hipoteze. V zadnjem poglavju podamo zaključek. V dodatkih podajamo še nekaj povezanih tematik s tem delom. Podamo statistično ovrednotenje naših algoritmov diferencialne evolucije in jih primerjamo s podobnimi algoritmi. Predlagamo rekonstrukcijo z večkriterijsko optimizacijo in povežemo animacijo ekosistema z evolucijskim modelom, ki vsebuje simulacije za oceno življenjskih pogojev. Nad evolucijskim modelom za animacijo ekosistema definiramo še interaktivno gojenje, kjer uporabnik med animacijo razvoja spreminja lastnosti evolucijskega modela. Nato sledi še seznam navedene literature, delovni življenjepis in priložene izjave.

2. Sorodna dela in ozadje

Smo ista vrsta, zvezdna stvar, nabirajoča zvezdno svetlobo. (Carl Sagan)

Za lažjo razlago in razumevanje predlaganih metod za potrditev postavljenih hipotez podamo v tem poglavju opis sorodnih del in iz svojega vidika razpravljamo o ozadju raziskovalnih področij, ki so vpeta v našem delu. Najprej predstavimo področje diferencialne evolucije, nadaljujemo z opisom proceduralnih drevesnih modelov in na koncu predstavimo še modeliranje dreves s pomočjo slik.

2.1 Diferencialna evolucija

Diferencialna evolucija (ang. *Differential Evolution* – DE) spada na področje računalništva in z njim povezane inteligence [225], kar bi lahko poimenovali tudi **umetna inteligenca** (ang. *Arfiticial Intelligence* - AI) [153]. Še natančneje lahko področje opredelimo v **računsko inteligenco** (ang. *Computational Intelligence* – CI) [67], ožje na področje **evolucijskega računanja** (ang. *Evolutionary Computation* – EC) [226, 78, 11, 77, 194, 95] in prvotno **matematičnega programiranja** (ang. *Mathematical Programming* – MP), t.j. matematične optimizacije [171].

V tem podpoglavju podajamo sorodna dela in ozadje algoritma diferencialne evolucije, od optimizacije in matematičnega programiranja, preko evolucijskih optimizacijskih algoritmov, definicije algoritma diferencialne evolucije do naših izboljšav algoritmov diferencialne evolucije. Podamo še načrt svojega novega algoritma za optimizacijo [264], predstavljenega v svoji magistrski nalogi [255], in podamo njegov psevdokod.

2.1.1 Optimizacija in matematično programiranje

Optimizacija (ang. *optimization*) je postopek povečevanja ustreznosti izraženosti pozitivnih lastnosti (kakovosti) rešitve zadanega problema, ki ga s tem imenujemo **optimizacijski problem** (ang. *optimization problem*). Optimizacijske probleme najdemo v najrazličnejših domenah stvarstva in nestvarstva z različnim številom lastnosti (pri merjenju, dimenzij), od obstajanja, trajanja, velikosti, številnosti, razporejenosti, relativnosti, omejitve, porazdelitve, usklajenosti, spremenljivosti, tendence, verjemljivosti, cene, zanesljivosti in drugih domenskih metrik ter pojmov[1].

[1]Epistemološki opis teh pojmov najdemo v enciklopedijah [253, 108].

Najstarejši optimizacijski problem, kot merimo čas [71, 93], je porazdelitev stvari in protistvari v sistemu obstoja časa, prostora in drugih dimenzij, ki poteka glede na stvarne in nestvarne vplive (fizikalne materijske in antimaterijske sile). Tisti del materije, ki ga trenutno sprejemamo kot živi [169] in je optimiziran znotraj tega sistema, pogosto definiramo s podsistemi atomov in molekul, ki se pojavljajo in z drugimi združujejo na tak način v organske sisteme, da opažen pojav pojmujemo kot **življenje** [68]. Primeri teh podsistemov smo živa bitja na Zemlji in primeri optimizacijskih problemov naše življenje kot celota, kot skupine ali kot posamezniki. Kot primer živih bitij na Zemlji smo človeška bitja (lat. *Homo sapiens sapiens* [141]) [48, 191], ljudje združeni v civilizacije [34].

Kot tudi druga živa bitja, je življenje ljudi spremljala množica izzivov vplivanja na dogodke in s tem spreminjanja v začetku omenjene ustreznosti, v tem primeru kakovosti življenja ljudi. Kot podmnožico teh izzivov lahko prepoznamo npr. vpliv na skupke stvari in interakcijo z njimi, kot je npr. uporaba materije za prehranjevanje, gibanje po prostoru, razmnoževanje in uporaba predmetnih orodij[2]. Skozi čas in prostor so ljudje[3] tako optimirali različna orodja za prehranjevanje, bivanje, umetnost in druge dejavnosti. Optimirani so bili postopki določitve dimenzij in izdelave orodij (npr. orožij), razporeditve predmetov (npr. gradnja bivalnih prostorov ali velikih monumentov, kot so piramide in zigurati [9] ter stolpnice [110]), dolžin, razmerij in površin prostora (npr. štiri tisočletja optimizacije točnega razmerja π [180, 170] ali zlato razmerje samopodobnosti ϕ [144]; oba uporabljana že tisočletja tudi v gradnji in slikarstvu [213]) ter merjenja časa (npr. s položaji nebesnih teles [79]).

S povečevanjem zahtevnosti rešenih nekaterih zgoraj navedenih izzivov se je povečevala tudi kompleksnost postopkov za določitev optimalnosti. Znanstveno področje optimizacije, ki obsega pripravo postopka za koračno izboljševanje optimalnosti rešitve, je matematično programiranje[4]. Enega prvih zapisanih optimizacijskih postopkov matematičnega programiranja je podal Kepler[5]. Newton čez nekaj desetletij kasneje predstavi račun, kalkulus [168], in z njegovo pomočjo poda gradientno metodo za pomikanje proti optimumu.

[2]Za definicijo orodja kot sredstva za preživetje oz. splošneje, povečanja ustreznosti, beri knjigo [54]

[3]Dosti krat lahko navedemo le določeno izvorno civilizacijo nekega izuma, saj nimamo beležk avtorstev in je izum toliko težje točno pripisati določenemu človeku.

[4]Pomen besede programiranje se v terminu navezuje na načrtovanje postopka, upravljanje, in ne na kodiranje s programskim jezikom, kot bi lahko morda pomislili [171].

[5]V svojem predlogu rešitve problema izbire soproge, ali, kot je kasneje formulacijo spremenil, osebne tajnice [116].

Ker jo bomo uporabljali tudi v nadaljevanju tega dela, s kalkulusno (računsko) kodo [168] definirajmo optimizacijski postopek in podajmo definicije nekaterih pojmov, kot so jih videli uporabniki tega, za računalniško programiranje [119], zgodnjega postopkovnega kodnega zapisa. Za razumevanje definicije optimizacijskega postopka najprej definirajmo problem, ki ga postopek rešuje. Optimizacijski problem definiramo kot iskanje dopustnega vektorja spremenljivk oziroma iskalnih parametrov $\mathbf{x} = (x_1, x_2, ..., x_D)$, ki optimira (minimira) vektorsko funkcijo $\mathbf{f}(\mathbf{x})$ z M kriteriji:

$$\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_M(\mathbf{x})). \tag{2.1}$$

Pri tem vsakemu vektorju spremenljivk $\mathbf{x}$ pripada njegov **kriterijski vektor** $\mathbf{f}(\mathbf{x})$, ki ga v prostoru kriterijev dobimo z ovrednotenjem rešitve iz prostora spremenljivk. Vektor $\mathbf{x}$ pri tem mora zadoščati m omejitvam:

$$g_i(\mathbf{x}) \geq 0; \quad i = 1, 2, ..., m, \tag{2.2}$$

s katerimi podamo omejitve v obliki več enokriterijskih funkcij. Tak vektor imenujemo dopustni vektor in vektor, ki krši omejitve, nedopustni vektor. Omejitve $g_i(\mathbf{x})$ lahko določajo npr. le definicijsko območje in zalogo vrednosti optimizirane funkcije:

$$\mathbf{g}(\mathbf{x}) \geq \mathbf{0} \Leftrightarrow (x_{\min,1}, x_{\max,1}, ..., x_{\min,D}, x_{\max,D}, f_{\min,1}, f_{\max,1}, ..., f_{\min,M}, f_{\max,M}). \tag{2.3}$$

Ob tem lahko povemo, da lahko vsako maksimizacijo kriterijev pretvorimo na problem minimizacije, tako da kriterije množimo z -1. Od tu naprej bomo na optimizacijo vedno gledali kot problem minimizacije.

V kolikor želimo najti optimalni vektor iz celotnega dopustnega območja, tak postopek imenujemo **globalna optimizacija** (ang. *global optimization* – GO). Področje matematičnega programiranja in posebej globalna optimizacija obsežnejši razmah doživi v času po drugi svetovni vojni, po predstavitvi digitalnih računalnikov, ki med drugim omogočijo avtomatsko izvajanje postopkov matematičnega programiranja [78]. Tukaj omenimo le algoritem preproste minimizacije s spuščanjem po hribu navzdol, Nelder-Mead [163], nadzorovano naključno preiskovanje [184], simulirano ohlajanje [115] in iskanje s tabuji [82].

Kasneje se pojavi še možnost numerične **večkriterijske optimizacije** (ang. *multi-objective optimization* – MO) [58]. Pri tej imamo možnost za izbiro najugodnejših rešitev problema najprej spoznati, kakšne možnosti sploh imamo, oz. kakšne najboljše možnosti imamo ter se šele kasneje odločiti za eno od njih. Namreč, ker so si kriteriji

pogosto v nasprotju, to pomeni, da izboljšanje neke odločitve po enem kriteriju povzroči njeno poslabšanje po drugih kriterijih. Takrat nimamo opravka samo z eno optimalno možno rešitvijo oz. odločitvijo, temveč z množico optimalnih rešitev. Ta pristop večkriterijske optimizacije, t.j. sočasne optimizacije medsebojno konfliktnih kriterijev, lahko uporabimo pri boljši podpori odločanju, saj odločanje zahteva izbiro ene izmed različnih danih možnosti, pogosto pa si ne želimo že v začetku postaviti prioritet za kriterije.

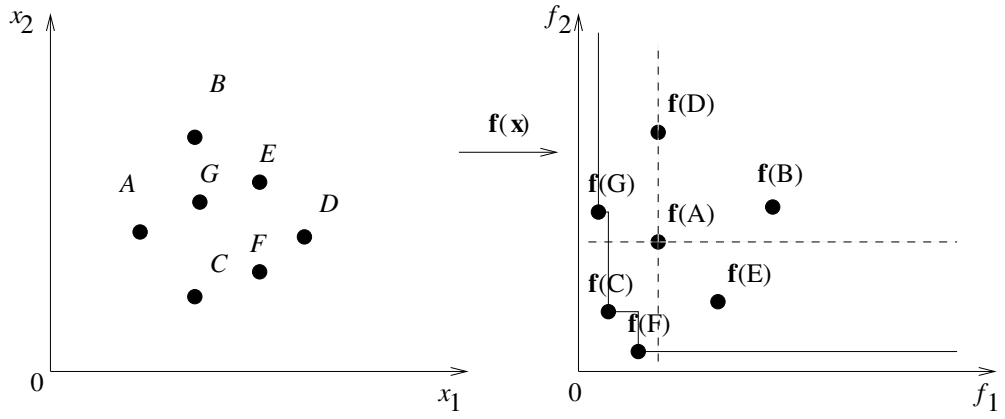
Definirajmo še relacijo **dominantnosti** (ang. dominance), označimo jo z $\mathbf{x} \preceq \mathbf{y}$. Vektor $\mathbf{x}$ dominira vektor $\mathbf{y}$, če velja:

$$f_i(\mathbf{x}) \leq f_i(\mathbf{y}),\ \forall\{i = 1, ..., M\}, \tag{2.4}$$

$$\exists j \in \{1, ..., M\} :\ f_j(\mathbf{x}) < f_j(\mathbf{y}), \tag{2.5}$$

torej da glede na izjavo 2.4 vektor $\mathbf{x}$ ni slabše ovrednoten od vektorja $\mathbf{y}$ po vseh kriterijih in glede na izjavo 2.5 je vektor $\mathbf{x}$ boljše ovrednoten od vektorja $\mathbf{y}$ po vsaj enem kriteriju.

Prostor kriterijev $\Re^M$ je z relacijo $\preceq$ delno urejen, saj nekateri vektorji medsebojno niso primerljivi. Slikovna razjasnitev pojma dominantnosti je vidna na sliki 2.1. Kot je videti iz slike, je vektor A boljše ovrednoten (boljši) od vektorjev B in D ter neprimerljiv (glede na dane kriterije) z E. Pri tem se neprimerljivost nanaša na nedominantnost pripadajočih kriterijskih vektorjev, saj primerjave njihovih komponent niso istoznačne.



Slika 2.1: Dvokriterijska funkcija $\mathbf{f}(\mathbf{x}) = (f_1(x_1, x_2), f_2(x_1, x_2))$ in dominantnost.

Pareto optimalni vektor je tak vektor $\mathbf{x}$, ki ga ne dominira noben drug dopustni vektor $\mathbf{z}$ in velja $\mathbf{f}(\mathbf{z}) \preceq \mathbf{f}(\mathbf{x})$. **Množica nedominiranih vektorjev** v množici vektorjev P je množica vseh tistih vektorjev, ki jih ne dominira noben vektor iz množice

P. Kriterijski vektorji teh vektorjev spremenljivk iz množice nedominiranih vektorjev tvorijo **nedominirano fronto**. Množico nedominiranih vektorjev celotnega prostora dopustnih vektorjev imenujemo **Pareto optimalna množica**, njene elemente **Pareto optimalni vektorji** in pripadajočo nedominirano fronto v prostoru kriterijev imenujemo **Pareto optimalna fronta**. Na sliki 2.1 tvorijo množico nedominiranih vektorjev vektorji G, C in F, saj niso dominiran od nobenega drugega vektorja, t.j. njihovi kriterijski vektorji $\mathbf{f}(\mathbf{G})$, $\mathbf{f}(\mathbf{C})$ in $\mathbf{f}(\mathbf{F})$ imajo manjše vrednosti od ostalih po vsaj enem kriteriju.

Pristope večkriterijskega optimiranja lahko, predvsem iz zgodovinskih razlogov [116, 44, 45], ločimo na dve vrsti. Prvi, t.i. **prednostni pristop**, večkriterijsko funkcijo pretvori v enokriterijsko (npr. s postavitvijo uteži w_i, $f(\mathbf{x}) = \sum_i w_i f_i(\mathbf{x})$). Reši enokriterijski pristop in postopek ponovi, z drugačno postavitvijo uteži (w_i). Če vnaprej poznamo ali določimo informacijo o pomembnosti kriterijev (metakriterij), nam to omogoči uporabo prednostnega pristopa. Sicer uporabimo drugi, t.i. **idealni pristop**, ki najprej poišče Pareto optimalno množico vektorjev. Nato iz nje sami izberemo vektor, ki nam najbolj ustreza. Izbiro najustreznejšega vektorja naredimo na podlagi dodatne informacije o pomembnosti kriterijev, ki navadno ni znana vnaprej in jo lahko izvemo šele po predstavitvi rezultatov optimizacije. Zato je idealni pristop bolj pregleden in manj subjektiven od prednostnega pristopa.

Pri optimizaciji z idealnim pristopom želimo, da večkriterijska optimizacijska metoda najde čim več Pareto optimalnih vektorjev. Ker izmed teh vektorjev (s pomočjo informacije o pomembnosti kriterijev) izbiramo najboljšega, si želimo, da so dobljeni vektorji kar se da enakomerno razporejeni po prostoru kriterijev (t.j. odločitvenem prostoru). Nalogo večkriterijskega optimiranja lahko tako pretvorimo v nalogo iskanja množice nedominiranih vektorjev, ki jo imenujemo **aproksimacijska množica** (ang. *approximation set*). Za elemente aproksimacijske množice želimo, da so čim bliže Pareto optimalni fronti in da so enakomerno razporejeni vzdolž Pareto optimalne fronte. Ti dve željeni lastnosti algoritmov za večkriterijsko optimizacijo si pogosto nasprotujeta. Zato je že sama zadostitev željenih lastnosti večkriterijske optimizacije lahko večkriterijska. Pogosto zanemarjena, tretja želena lastnost algoritmov večkriterijske optimizacije je čim krajši čas računanja za doseg prvih dveh ciljev večkriterijske optimizacije [57]. Primer zmanjšanja časa računanja je uporaba superskalarnih arhitektur, kot je CUDA, na kateri smo predstavili primer izračunavanja enega od realnih industrijskih problemov [272]. Z algoritmom večkriterijske optimizacije želimo doseči tudi

čim večjo robustnost ob različnih zagonih algoritma [56, 70], t.j. da algoritem s čim večjo verjetnostjo uspešno optimizira dan problem.

Omenimo nekaj prvih metod za optimizacijo, kodiranih s kalkulusom. Oglejmo si osnovno Newtonovo metodo in dve najbolj priljubljeni osnovni metodi za večkriterijsko optimizacijo: metodo utežene vsote in metodo ϵ-omejitev. Več o osnovnih metodah najdemo v [58].

Newton je za funkcijo podal izračun gradienta, s pomočjo simboličnega ali numeričnega odvoda funkcije:

$$f'(x) = \frac{\Delta x}{\Delta f(x)}, \tag{2.6}$$

kjer Δx označuje spremembo po spremenljivki x in $\Delta f(x)$ pripadajočo spremembo funkcijske vrednosti. Kasneje Newtonovo metodo dopolni Lagrange, v postopku za določitev optimuma, tako da gradient prišteje, integrira, k izhodiščni točki:

$$f^*(x) = f(x) + \Delta x f'(x), \tag{2.7}$$

pri čemer $f^*(x)$ označuje optimiran približek in $f'(x)$ diferencial oz. spremembo funkcije $f(x)$ za interval Δx.

Metoda utežene vsote (ang. *weighted sum method*) je najbolj razširjena klasična metoda za večkriterijsko optimiranje. Večkriterijski problem pretvorimo v enokriterijskega tako, da izberemo uteži w_i, ki določajo pomembnost kriterijev. Tako iz naloge

$$\text{optimiraj } \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), ..., f_M(\mathbf{x})) \tag{2.8}$$

dobimo nalogo:

$$\text{optimiraj } \mathbf{g}(\mathbf{x}) = (w_1 f_1(\mathbf{x}) + ... + w_M f_M(\mathbf{x})), \tag{2.9}$$

ki jo rešimo z eno od metod za enokriterijsko optimiziranje. Ker se optimum te naloge ne spremeni, če uteži pomnožimo s konstanto, navadno predpostavljamo, da zanje velja

$$w_i \in [0, 1] \text{ in } \sum_{i=1}^{M} w_i = 1. \tag{2.10}$$

Največja prednost te metode je njena enostavnost. Če rešujemo konveksen večkriterijski optimizacijski problem, pri katerem so prednosti kriterijev poznane, je metoda uteženih vsot prava izbira. Metoda pa ima več slabosti. Prva očitna slabost te metode je lahko, da zahteva vektor uteži. Če ne poznamo prednosti kriterijev, je določitev takšnega vektorja zahtevna naloga. Poleg tega moramo, da bi dobili več Pareto optimalnih

vektorjev, metodo uporabiti večkrat – vsakič z drugačno nastavitvijo uteži. Vendar je pri tem treba paziti, saj z enakomerno porazdelitvijo vektorjev uteži ne najdemo nujno enakomerno porazdeljenih kriterijskih vektorjev na Pareto optimalni fronti.

Ker nam odvisnost med vektorjem uteži in vektorji na Pareto optimalni fronti navadno ni poznana, težko postavimo nastavitev uteži, ki bo našla željen kriterijski vektor Pareto optimalne fronte. Poleg tega različni vektorji uteži ne najdejo nujno različnih kriterijskih vektorjev na Pareto optimalni fronti. Podobno tudi en vektor uteži lahko vrne različne kriterijske vektorje Pareto optimalne fronte. Če je večkriterijski optimizacijski problem konveksen, potem lahko vsako točko Pareto optimalne fronte izračunamo z metodo utežene vsote. Vendar je metoda omejena na konveksne Pareto optimalne fronte, za nekonveksne fronte pa odpove. Metoda je tudi občutljiva na razmerja med vrednostmi kriterijev, zato je dobro te vrednosti predhodno normirati.

Če se želimo izogniti težavam, ki so značilne za metodo utežene vsote pri nekonveksnih Pareto optimalnih frontah, lahko uporabimo **metodo ϵ-omejitev** (*ϵ-constraint method*). Predlagali so jo Haimes in sodelavci [88] in rešuje večkriterijski optimizacijski problem tako, da optimira samo en kriterij, ostale pa doda med omejitve. Preoblikovana optimizacijska naloga se tako glasi:

$$\text{optimiraj } f_j(\mathbf{x}) \text{ pri pogojih } f_i(\mathbf{x}) \leq \epsilon_i, \quad i = 1, ..., M \text{ in } i \neq j. \tag{2.11}$$

V tej obliki parameter ϵ_i označuje zgornjo mejo za vrednost f_i in ne pomeni nujno majhne vrednosti blizu nič. Kot zanimivost lahko povemo še, da se metoda ϵ-omejitev uspešno uporablja tudi na področju enokriterijske optimizacije z omejitvami [217], za podoben namen, kako pretvoriti enokriterijski problem s podanimi omejitvami v enokriterijskega brez omejitev.

Podobno kot metoda utežene vsote tudi metoda ϵ-omejitev zahteva informacijo o pomembnosti kriterijev. Tokrat je ta podana v obliki vektorja omejitev. Če izberemo drugačen vektor omejitev ali če za optimiziranje izberemo drug kriterij, kot rezultat dobimo drugačno množico kriterijskih vektorjev. Ker omejujemo kriterije (in ne spremenljivk), nam ta metoda omogoča lažje lokaliziranje rešitev v željenih območjih. Poleg tega nima težav z obliko Pareto optimalne fronte, saj deluje na enak način za konveksne in nekonveksne ter zvezne in diskretne fronte. Metoda tudi ne zahteva normiranja vrednosti ovrednotenj kriterijev, saj razlike v vrednosti kriterijev upoštevamo v vektorju omejitev.

Če želimo v optimizaciji posebej obravnavati še kriterijske omejitve, imenujemo takšno optimizacijo, **optimizacija z omejitvami** (ang. *constraint optimization*). Skozi zgodovino najdemo definicijo številnih metod za obravnavo omejitev v optimizacijskih algoritmih. Michalewicz in sod. [160, 124] so te metode uvrstili v štiri skupine:

1. *Metode z ohranjanjem dopustnih rešitev.* Ta skupina metod zajema posebne operatorje, ki dopustne vektorje staršev vedno preslika v dopustne nove vektorje potomcev. Začetna populacija mora vsebovati le dopustne vektorje in omejitve so lahko le linearne.

2. *Metode s kazensko funkcijo.* Pri tej skupini metod z zunanjo kazensko funkcijo ocenimo kazen nedopustnih rešitev. Metode znotraj te skupine se razlikujejo v podrobnostih, kako pripraviti kazensko funkcijo in kako jo uporabiti pri kaznovanju nedopustnih posameznikov.

3. *Metode s strogim razlikovanjem med dopustnimi in nedopustnimi vektorji.* Metode v tej skupini strogo ločijo med dopustnimi in nedopustnimi vektorji. Primer takšne metode je ločitev vsakega dopustnega vektorja $\mathbf{x}$ in vsakega nedopustnega vektorja $\mathbf{y}$ s strogo urejenostjo: $\mathbf{f}(\mathbf{x}) \preceq \mathbf{f}(\mathbf{y})$, t.j. vsak dopustni vektor je boljši od vsakega nedopustnega vektorja.

4. *Druge mešane metode.* Tu združimo metode iz več področij.

Večino problemov z omejitvami je moč obravnavati z metodami s kazensko funkcijo. Funkcija meri oceno kršitve omejitev, ki so lahko enakostne ali neenakostne. Vektor $\mathbf{x}$ je *dopusten* za neenakostne omejitve $g_i(\mathbf{x}) \leq 0$ in enakostne omejitve $h_j(\mathbf{x}) = 0$, če:

$$g_i(\mathbf{x}) \leq 0, \qquad i = 1, ..., q, \tag{2.12}$$

$$|h_j(\mathbf{x})| - \epsilon \leq 0, \qquad j = q+1, ..., m, \tag{2.13}$$

kjer enakostne omejitve pred oceno dopustnosti pretvorimo v neenakostne omejitve za nek interval ϵ, npr. $\epsilon = 0,0001$ za funkcije [137]. Povprečno vrednost kršenja vseh omejitev $\overline{v}$ lahko definiramo kot [137]:

$$\overline{v} = \frac{(\sum_{i=1}^{q} G_i(\mathbf{x}) + \sum_{j=q+1}^{m} H_j(\mathbf{x}))}{m}, \tag{2.14}$$

kjer

$$G_i(\mathbf{x}) = \begin{cases} g_i(\mathbf{x}), & g_i(\mathbf{x}) > 0, \\ 0, & g_i(\mathbf{x}) \leq 0, \end{cases} \tag{2.15}$$

$$H_j(\mathbf{x}) = \begin{cases} |h_j(\mathbf{x})|, & |h_j(\mathbf{x})| - \epsilon > 0, \\ 0, & |h_j(\mathbf{x})| - \epsilon \leq 0. \end{cases} \tag{2.16}$$

Ta kriterij nam pravi, da je vsota kršitev vseh omejitev enaka nič za dopustne vektorje in večja od nič, ko je kršena vsaj ena omejitev. Da bi našli dopustne vektorje, ta kriterij uporabimo za vodenje iskanja proti območjem dopustnih vektorjev. Implementacije tega vodenja v optimizacijskih algoritmih zajemajo obsežen nabor idej, povzetek pristopov najdemo v [159, 43].

2.1.2 Matematično programiranje in evolucijsko računanje

V 19. stoletju anno domini (AD) se v raziskavah iz bioloških znanosti pojavi vseprisotna evolucijska teorija darvinizma [105] in nato z njo vezan neodarvinizem [202], ki poleg Darwinovega zakona naravne selekcije [48] vsebuje še Weismannovo teorijo dedne plazme [239] in Mendlove zakone dedovanja [155]. Izraz neodarvinizem je bil tudi še kasneje populariziran [54] v namene sklicevanja na moderno evolucijsko sintezo [103]. Teorije o evoluciji oz. vsaj potomstvu so se dopolnjevale tisočletja in pravkar citirani pristopi zajemajo zgolj bolj citirana dela, medtem ko obstajajo še številne druge (npr. predhodna E. Darwina [50] in tudi analogna A. R. Wallacea [235, 49] leto pred objavo [48]). Obstajajo tudi naravne teorije iz antike in še starejše, ki nastanek življenja omejijo zgolj na kreacionizem, najpogosteje iz vode kot generativnega elementa, z načrtovano pojavitvijo po vodni poplavi [222, 199, 92, 162, 39], včasih ex nihilo [36, 199, 92], ki pa so precej manj podrobne, bolj skope z empiričnimi dokazi in pogosto v nasprotju s prej omenjenimi. Pojem neodarvinizma z modelom odmisli (t.j. metodično abstrahira [94]) dinamiko življenja zgolj na naslednje mehanizme [69, 156]:

- **reprodukcija** (ang. *reproduction*): način, kako iz obstoječega nabora primerkov narediti nove,
- **mutacija** (ang. *mutation*): manjša sprememba kodirane dedne zasnove, ki povzroči večje spremembe v obnašanju primerka,
- **tekmovanje** (ang. *competition*): mehanizem za ovrednotenje ocene ustreznosti primerka v danem okolju in
- **selekcija** (ang. *selection*): vzdržuje ali povečuje ustreznost populacije, ustreznost definira kot sposobnost preživetja in reprodukcije v danem okolju.

Evolucijska ideologija v 20. stoletju vpliva na področje matematičnega programiranja tako, da znotraj nastajajočega skupka algoritmov umetne inteligence [225, 153], nastane skupno področje evolucijskega računanja [226, 78, 11, 77, 194, 95]. Evolucijsko računanje je tako področje algoritmov, ki po evolucijskem vzoru simulirajo nek naravni proces. Ta naravni proces je abstrahiran v model, vsebovan v posameznem algoritmu [76]. Večina teh modelov temelji na pojmu neodarvinizma [104], ki je tako njihova biološka podlaga. Modeli so najpogosteje definirani na treh stopnjah podrobnosti: najprej opisani domensko prozaično (t.j. za opis so uporabljeni pojmi določene

domene iz narave), nato zapisani v kalkulusu (t.j. definirane so še računske operacije med pojmi) in še kodirani v programskem jeziku (t.j. definirani s tipiziranimi programskimi stavki in strukturami). Te tri stopnje podrobnosti nam dajo vsaj tri izhodišča, kako narediti primerjavo podobnosti modelov, kje med njimi ločiti in v kakšne vrste jih lahko razdelimo.

Glede na domensko prozaične definicije njihovih modelov, ločimo več vrst algoritmov iz nabora evolucijskega računanja in ti pogosto služijo za optimizacijo. Neketere optimizacijske algoritme znotraj evolucijskega računanja lahko po dogovorjenem izboru pojmovno označimo tudi z imenom **evolucijski algoritmi** (ang. *evolutionary algorithms*). Primeri nekaterih algoritmov iz omenjenega nabora so:

- **genetski algoritem** (ang. *Genetic Algorithm* – GA) [85] posameznike izdeluje tako, da realna števila v posameznem vektorju kodira z binarnim nizom, nato z operatorji mutacije in križanja spreminja posamezne bite v binarni predstavitvi. Operator mutacije navadno deluje uniformno naključno nad vsakim bitom binarno kodiranega posameznika. Operator križanja združi posamezne bite iz več posameznikov (staršev) v novega posameznika, tako da jemlje nekaj bitov iz vsakega od staršev,

- **genetsko programiranje** (ang. *Genetic Programming* – GP) [123, 132] gradi drevesno strukturo, kot jo pozna programski jezik lisp [152], program; uporabljeni genetski operatorji so posebej prirejeni za operacije nad drevesom,

- **evolucijsko programiranje** (ang. *Evolutionary Programming* – EP) uporablja končni avtomat za predstavitev in je bilo predstavljeno kot pristop v umetni inteligenci [77], enega od kasnejših algoritmov pa predstavlja delo [248], ki daje tudi primerjalno osnovo za številne medsebojne primerjave ostalih algoritmov nad standardnim izbranim naborom testnih funkcij za enokriterijsko optimizacijo,

- **memetski algoritmi** (ang. *Memetic Algorithms* – MA) združujejo evolucijske pristope s pristopi iskustveno prilagojenih (hevrističnih) algoritmov za reševanje ožje določenih optimizacijskih problemov; novejši pregled področja zajema [164],

- **evolucijske strategije** (ang. *Evolution Strategies* – ES), ki iskalne parametre kodirajo z realnimi vektorji, v posameznem vektorju pa kodirajo tudi krmilne parametre iskalnega algoritma (pregled podan v [14, 8, 207], močnejši predstavniki skupine so [91, 107, 81]),

- **umetni imunski sistemi** (ang. *Artificial Immune Systems* – AIS), ki uporabljajo vzor imunskih sistemov za iskanje rešitve proti sovražnim telesom [114],
- **rojna inteligenca** (ang. *Swarm Intelligence* – SI), ki uporablja nek altruistični vzor iz rojev ali kolonij s skupno inteligenco, ki se pojavi ob sodelovanju in izmenjavi med posamezniki [13]; sledi nekaj primerov SI,
- **optimizacija z rojem delcev** (ang. *Particle Swarm Optimization* – PSO) poleg vsakega vektorja spremenljivk (delca) hrani še njegov diferenčni vektor, ki modelira hitrost premikanja delca v geografskem (kriterijskem) prostoru [113, 283, 281],
- **optimizacija s kolonijo mravelj** (ang. *Ant Colony Optimization* – ACO) modelira pomnjenje zgodovine dobrih iskalnih parametrov po vzoru mravelj, ki v primeru najdene hrane za seboj puščajo feromonske sledi [66, 122, 24],
- **optimizacija s čebelami** (ang. *Bees Algorithm*), ki posnema delovanje čebel pri nabiranju hrane [177]; daljšo razlago in preskus algoritma najdemo v diplomski nalogi [19],
- **optimizacija s kolonijami čebel** (ang. *Artificial Bee Colony* – ABC), ki temelji na vzorih algoritma za optimizacijo s čebelami in dodaja razširitev s kolonijami [112],
- **kukavičje iskanje** (ang. *Cuckoo Search* – CS), ki posnema parazitstvo valjenja kukavičjih zarodkov v tujih gnezdih [245],
- **bakterijsko preiskovanje** (ang. *Bacterial Foraging Optimization – BFO*), ki posnema iskanje hrane po vzoru bakterij *E. coli* [176],
- **umetne travne kolonije** (ang. *Artificial Weed Colonies* – AWD), ki posnema rast trave; algoritem so primerjali tudi z enim od naših v prispevku [128],
- **diferencialna evolucija** (ang. *Differential Evolution* – DE), ki združuje simpleks metodo Nelder-Mead [163], nadzorovano naključno preiskovanje [184] in nekatere evolucijske mehanizme ter je opisana v naslednjem razdelku in še
- nabor drugih algoritmov znotraj evolucijskega računanja, kot so **harmonično iskanje** (ang. *Harmony Search* – HS) [80], **učljiv evolucijski model** (ang. *Learnable Evolution Model – LEM*) [241] in **algoritem kresničke** (ang. *Firefly Algorithm* – FA) [244], in ti algoritmi povečini vsebujejo vsaj nekaj gradnikov EA.

2.1.3 Algoritem diferencialne evolucije

Diferencialna evolucija (DE) [215] je **populacijski algoritem**, ki se uspešno uporablja za globalno optimizacijo **realno kodiranih** [84] numeričnih funkcij. Algoritem zaradi svoje narave prilagajanja optimizacijskega koraka (pri tvorbi potomcev se upoštevajo diference med starši) in **stabilnega iskanja** (ang. *steady-state search*) z elitističnim selekcijskim mehanizmom navadno daje boljše rezultate od drugih evolucijskih algoritmov [23, 157, 166, 52]. Algoritem diferencialne evolucije [215] sestoji iz glavne evolucijske zanke, v kateri z evolucijskimi operatorji mutacije, križanja in selekcije postopno in vzporedno po generacijah izboljšuje približek iskane rešitve. Evolucijski operatorji vplivajo na vsak primerek $\mathbf{x}_i$, $\forall i \in \{0, 1, ..., NP-1\}$ v populaciji rešitev, iz katerih se zgradi nova populacija. Posameznike po tem ovrednotimo, skupno število ovrednotenj posameznikov pa označimo s FEs. Algoritem DE je tudi preprost in ima malo parametrov, F – **faktor skaliranja** diferenčnega vektorja za mutacijo, CR – **stopnja križanja**, NP – **velikost populacije** ter včasih tudi s – **strategija** mutacije in križanja.

Natančneje razložimo osnovni algoritem diferencialne evolucije, ki deluje za enokriterijsko optimizacijo [215]. DE sestoji iz glavne evolucijske zanke, v kateri z evolucijskimi operatorji mutacije, križanja in selekcije postopno in vzporedno izboljšuje približek iskane rešitve. Algoritem DE ima globalno in lokalno povezano velikost diferenčnega koraka, ki se samoprilagaja skozi čas glede na položaj posameznikov populacije v iskalnem prostoru [109]. Evolucijski operatorji vplivajo na vsak primerek $\mathbf{x}_i$, $\forall i \in \{0, 1, ..., NP-1\}$ v populaciji rešitev, iz katerih se zgradi nova populacija za naslednjo **generacijo** (ang. *generation*). Eno izdelavo novega posameznika poimenujemo **iteracija** (ang. *iteration*). V vsaki iteraciji operator mutacije izračuna **mutiran vektor** (ang. *donor vector*) $\mathbf{v}_{i,g+1}$:

$$\mathbf{v}_{i,g+1} = \mathbf{x}_{r_1,g} + F \times (\mathbf{x}_{r_2,g} - \mathbf{x}_{r_3,g}), \tag{2.17}$$

kjer so $r_1, r_2, r_3 \in \{0, 1, ..., NP-1\}$ paroma in od i različni indeksi primerkov iz populacije v generaciji $g \in \{0, 1, \ldots, G-1\}$, $i \in \{0, 1, ..., NP-1\}$ in $F \in [0, 2]$. Vektor r_1 imenujemo **osnovni vektor** (ang. *base vector*). Izraz $\mathbf{x}_{r_2,G} - \mathbf{x}_{r_3,G}$ imenujemo **diferenčni vektor** (ang. *difference vector*) in po množenju s faktorjem ojačanja F, **utežen diferenčni vektor** (ang. *weighted difference vector*). Pravkar opisano strategijo mutacije imenujemo rand/1, obstajajo še druge [23], kot je strategija best/1:

$$\mathbf{v}_{i,g+1} = \mathbf{x}_{\text{best},g} + F \times (\mathbf{x}_{r_1,g} - \mathbf{x}_{r_2,g}), \tag{2.18}$$

kjer $\mathbf{x}_{\text{best},g}$ označuje najbolje ocenjen primerek iz generacije g. V mutacijo lahko vključimo tudi starševski primerek $\mathbf{x}_{i,g}$, kot v strategiji rand-to-best/1:

$$\mathbf{v}_{i,g+1} = \mathbf{x}_{i,g} + F \times (\mathbf{x}_{\text{best},g} - \mathbf{x}_{i,g}) + F \times (\mathbf{x}_{r_1,g} - \mathbf{x}_{r_2,g}). \tag{2.19}$$

V kolikor v mutacijo vključimo več naključnih posameznikov, poznamo še strategijo rand/2:

$$\mathbf{v}_{i,g+1} = \mathbf{x}_{r_1,g} + F \times (\mathbf{x}_{r_2,g} - \mathbf{x}_{r_3,g}) + F \times (\mathbf{x}_{r_4,g} - \mathbf{x}_{r_5,g}) \tag{2.20}$$

in strategijo mutacije z najboljšim vektorjem, best/2:

$$\mathbf{v}_{i,g+1} = \mathbf{x}_{\text{best},g} + F \times (\mathbf{x}_{r_1,g} - \mathbf{x}_{r_2,g}) + F \times (\mathbf{x}_{r_3,g} - \mathbf{x}_{r_4,g}). \tag{2.21}$$

Po mutaciji dobljeni mutiran vektor $\mathbf{v}_{i,g+1}$ križamo s **ciljnim vektorjem** (ang. *target vector*) $\mathbf{x}_{i,g}$ in tako dobimo **poskusni vektor** (ang. *trial vector*) $\mathbf{u}_{i,g+1}$. Operator križanja v algoritmu DE prevzema dve obliki, imenovani binarno križanje (/bin) ali eksponentno križanje (/exp). Prvo zapišemo kot:

$$u_{i,j,g+1} = \begin{cases} v_{i,j,g+1} & \text{če rand}(0,1) \leq CR \text{ ali } j = j_{\text{rand}} \\ x_{i,j,g} & \text{sicer} \end{cases}, \tag{2.22}$$

kjer $j \in \{1, ..., D\}$ označuje j-to komponento vektorja v prostoru z D dimenzijami, funkcija $\text{rand}(0,1) \in [0,1]$ označuje vzorčenje uniformno (psevdo) naključno porazdeljenega števila in j_{rand} izbira uniformno naključen indeks iskalnega parametra, ki ga vedno izmenjamo (da bi s tem preprečili izdelavo enakih posameznikov). CR označuje že omenjen krmilni parameter stopnje križanja.

Eksponentno križanje lahko izrazimo kot prepis le določenih zaporednih iskalnih parametrov:

$$u_{i,j,g+1} = \begin{cases} v_{i,j,g+1} & \text{če } j = \langle n \rangle_D, \langle n+1 \rangle_D, ..., \langle n+l-1 \rangle_D \\ x_{i,j,g} & \text{sicer} \end{cases}, \tag{2.23}$$

kjer $\langle \rangle_D$ označuje operacijo ostanka pri deljenju z D, n označuje indeks začetka iz $v_{i,j,g+1}$ prepisovanega zaporedja in $l \in [0, D-1]$ njeno eksponentno porazdeljeno dolžino zaporedja $P(l = \upsilon) = (CR)^{\upsilon}$.

Operator selekcije v algoritmu DE je dokaj preprost, saj za vsak novo generiran primerek zgolj preveri, ali je ocena $f(\mathbf{u}_{i,g+1})$ poskusnega vektorja ustreznejša (oz. boljša, saj boljše pomeni manjšo vrednost) od ocene ciljnega vektorja $f(\mathbf{x}_{i,g})$:

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g+1} & \text{če } f(\mathbf{u}_{i,g+1}) < f(\mathbf{x}_{i,g}) \\ \mathbf{x}_{i,g} & \text{sicer} \end{cases}, \tag{2.24}$$

Tabela 2.1: Diferencialna evolucija – kombinacije operatorjev križanja in mutacije ter njihovo poimenovanje v implementaciji [182].

1	DE/best/1/exp
2	DE/rand/1/exp
3	DE/rand-to-best/1/exp
4	DE/best/2/exp
5	DE/rand/2/exp
6	DE/best/1/bin
7	**DE/rand/1/bin**
8	DE/rand-to-best/1/bin
9	DE/best/2/bin
10	DE/rand/2/bin

kjer algoritem DE privzema elitističen princip preživetja najustreznejših [48].

S kombinacijo naštetih operatorjev mutacije in križanja se je z objavo implementacije algoritma DE [182] pojavilo oštevilčenje imen teh kombinacij v algoritmu DE, ki jih vidimo v tabeli 2.1. V kolikor imamo iskalni problem, kjer potrebujemo hitro konvergenco, lahko izberemo strategijo 1 ali strategije 4, 6 in 9. Najpogosteje uporabljena je strategija DE/rand/1/bin, ki jo prikazuje algoritem 1. Strategije 2, 5 in 10 lahko uporabimo, če ostale odpovejo, obstaja pa tudi veliko število izboljšav in dopolnitev osnovnega algoritma diferencialne evolucije, ki jih navajamo v nadaljevanju.

Algoritem 1 Osnovni algoritem DE [215] s strategijo rand/1/bin.

Vhod: $\mathbf{f}(\mathbf{x})$ – kriterijska funkcija; D, NP, G – krmilni parametri DE

Izhod: $\mathbf{x}_{\text{best}}$ – optimirani parametri za dano kriterijsko funkcijo

1: `Uniformno naključno generiraj začetno populacijo DE` $\mathbf{x}_{i,0}$ `za` $i = 1..NP$;
2: **for** DE generacijska zanka g (dokler $g < G$) **do**
3: **for** DE iteracijska zanka i (za vse vektorje $\mathbf{x}_{i,g}$ v trenutni populaciji) **do**
4: DE izračun poskusnega vektorja $\mathbf{x}_{i,g}$ (mutacija, križanje):
5: $$\mathbf{v}_{i,g+1} = \mathbf{x}_{r_1,g} + F \times (\mathbf{x}_{r_2,g} - \mathbf{x}_{r_3,g});$$
6: $$u_{i,j,g+1} = \begin{cases} v_{i,j,g+1} & \text{če } rand(0,1) \leq CR \text{ ali } j = j_{\text{rand}} \\ x_{i,j,g} & \text{sicer} \end{cases};$$
7: DE selekcija z oceno ustreznosti $f(\mathbf{u}_{i,G+1})$:
8: $$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g+1} & \text{če } f(\mathbf{u}_{i,g+1}) < f(\mathbf{x}_{i,g}) \\ \mathbf{x}_{i,g} & \text{sicer} \end{cases};$$
9: **end for**
10: **end for**
11: **vrni** najboljši najden posameznik ($\mathbf{x}_{\text{best}}$);

Osnovni algoritem diferencialne evolucije ima številne razširitve. Že Lampinen je zbral tedaj obsežno zbirko izboljšav algoritma DE [129]. Sedaj obstaja precej izboljšav osnovnega algoritma DE, omenimo le nekaj pomembnejših:

- DE s trigonometrično mutacijo avtorjev Fan in Lampinen [73], ki je posebej primeren za reševanje rotiranih problemov,
- samoprilagodljivi DE avtorjev Liang, Yao in Newton [138],
- večkriterijski DE avtorjev Xue, Sanderson in Graves [243],
- DE z mehko mutacijo avtorjev Liu in Lampinen [143],
- samoprilagodljivi DE avtorjev Omran, Salman in Engelbrecht [172],
- DE avtorjev Qin in Suganthan [189],
- DE s prilagajanjem dopustnosti za optimizacijo z omejitvami avtorjev Takahama, Sakai in Iwane [217],
- DE s prilagajanjem velikosti populacije avtorja J. Teo [221],
- DE s prilagajanjem krmilnega parametra CR avtorja M. M. Ali [4],
- DE s samoprilagajanjem krmilnih parametrov F in CR avtorjev Huang, Qin in Suganthan [99],
- DE z logističnim samoprilagajanjem krmilnih parametrov F in CR (jDE) avtorjev Brest s sod. [23, 26], ki je skupaj s svojimi izboljšavami opisan kasneje v nadaljevanju tega poglavja,
- DE s prilagajanjem izbrane mutacijske strategije avtorjev Qin, Huang in Suganthan [188],
- DE za optimizacijo problemov z velikim številom dimenzij avtorjev Yang, Tang in Yao [247],
- DE z novim logističnim samoprilagajanjem krmilnih parametrov F in CR in spremenjeno mutacijsko strategijo (JADE) avtorjev Zhang in Sanderson [275, 274],
- samoprilagodljivi DE s preiskovanjem okolice avtorjev Yang, He in Yao [246],

- naš DE z normalno porazdeljenim samoprilagajanjem krmilnih parametrov F in CR po vzoru evolucijskih strategij, SA-DE [31], in DEwSAcc s koevolucijo [263],
- memetski DE avtorjev Caponio, Neri in Tirronen [37],
- DE s podpopulacijami z indeksnim sosedstvom avtorjev Das, Abraham, Chakraborty in Konar [51],
- DE z ansambli strategij avtorjev Yu in Suganthan [250] in ponovno Mallipeddi, Suganthan, Pan ter Tasgetiren [147],
- DE za manjše kompaktne sisteme avtorjev Mininno, Neri, Cupertino in Naso [161],
- ter študije posebej parametriziranih DE, avtorjev D. Zaharie [251], J. Tvrdik [230] in zbrani nabori mutacijskih strategij DE avtorjev Price in Storn [183], knjiga V. Feoktistov [74], knjiga D. B. Fogel [76], pregledih področja avtorjev Montes s sod. [158, 157], Neri in Tirronen [166] in predvsem Das in Suganthan [52]. Pri tem lahko dodamo še, da se več omenjenih avtorjev s citatom sklicuje tudi na katerega od naših prispevkov.

Pri algoritmu DE je bila potrjena tudi potreba po samoprilagajanju krmilnih parametrov [138, 143, 172] in uporabljena v [189, 221] in kasneje npr. [23, 26, 22, 27, 21]. Samoprilagajanje krmilnih parametrov v evolucijskih algoritmih v splošnem izhaja iz evolucijskih strategij [14, 8, 207], kjer so krmilni parametri vključeni v posameznike skupaj z iskalnimi parametri. Notacija splošne evolucijske strategije (ES) je $\mu/\rho, \lambda$-ES, kjer je μ velikost starševske populacije, ρ število staršev za vsakega novega posameznika in λ velikost nove populacije. Posameznik je definiran kot $\mathbf{p} = (\mathbf{x}, \mathbf{s}, F(\mathbf{x}))$, kjer so $\mathbf{x}$ iskalni parametri, $\mathbf{s}$ krmilni parametri in $F(\mathbf{x})$ ocenitev posameznika. Čeprav obstajajo druge notacije in variante ES [91, 90, 107], omenimo zaradi umestitve DE zaenkrat zgolj osnovno varianto ES. Ta uporablja logaritmično normalno porazdeljeno [139] prilagajanje krmilnih parametrov, ki se skozi selekcijo samoprilagajajo. Preživijo le posamezniki, ki se prilagajajo dobrim krmilnim parametrom in dobrim iskalnim parametrom. Z bolj primernimi vrednostmi krmilnih parametrov proces iskanja namreč doseže boljše rešitve in zaradi tega se iskanje hitreje približuje boljšim rešitvam, ki preživijo z večjo verjetnostjo in ustvarijo več potomcev ter s tem razširijo svoje krmilne parametre [70, 69].

Poznamo več različic algoritma DE s samoprilagajanjem krmilnih parametrov, izmed katerih natančneje opišimo algoritem jDE [23], ki ga prikazuje algoritem 2. Algoritem jDE vključuje mehanizem samoprilagajanja krmilnih parametrov F in CR, ki so ga uvedli Brest s sod. [23]. Slednji mehanizem uporablja zgoraj predstavljeno strategijo 'rand/1/bin'. Mehanizem iz [23] samoprilagaja krmilna parametra F in CR med evolucijskim procesom; NP ostaja nespremenjen. Vsak posameznik v populaciji je bil razširjen z vrednostmi teh dveh samoprilagodljivih krmilnih parametrov. Njuni vrednosti se spreminjata:

$$F_{i,g+1} = \begin{cases} F_{\mathrm{l}} + rand_1 \times F_{\mathrm{u}} & \text{če } rand_2 < \tau_1, \\ F_{i,g} & \text{sicer,} \end{cases} \tag{2.25}$$

$$CR_{i,g+1} = \begin{cases} rand_3 & \text{če } rand_4 < \tau_2, \\ CR_{i,g} & \text{sicer.} \end{cases} \tag{2.26}$$

Novi vrednosti za F in CR sta shranjeni v nov vektor. $rand_j, j \in \{1, 2, 3, 4\}$ so uniformno naključna števila $\in [0, 1]$. τ_1 in τ_2 predstavlja verjetnost prilagajanja obeh krmilnih parametrov F in CR. $\tau_1, \tau_2, F_{\mathrm{l}}, F_{\mathrm{u}}$ imajo nespremenljive vrednosti 0,1, 0,1, 0,1, 0,9. Nov F zavzema naključne vrednosti med [0,1, 1,0] in nov CR med $[0, 1]$. $F_{i,g+1}$ in $CR_{i,g+1}$ sta izračunana pred mutacijo, po vzoru iz izkušenj ob razvoju ES. Nova parametra tako vplivata na mutacijo, križanje in selekcijo novega vektorja $\mathbf{x}_{i,g+1}$.

Ena od razširitev algoritma jDE je tudi obravnava omejitev, algoritem ϵ-jDE [20], ki sloni na osnovnem algoritmu jDE [23] in prvem dopolnjenem algoritmu za obravnavo omejitev z jDE, jDE-2 [26]. Algoritem ϵ-jDE omejitve obravnava v selekciji, tako da prilagojeno izbira med vektorjema z indeksi i in j:

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{x}_{j,g} & \text{če } (\overline{v}_{i,g} > \overline{v}_{j,g}), \\ \mathbf{x}_{j,g} & \text{sicer če } (\overline{v}_{j,g} = 0) \wedge (f(\mathbf{x}_{j,g}) < f(\mathbf{x}_{i,g})), \\ \mathbf{x}_{i,g} & \text{sicer.} \end{cases} \tag{2.27}$$

Algoritem razlikuje med dopustnimi ($\overline{v} = 0$) in nedopustnimi vektorji: vsak dopustni vektor je boljši od vsakega nedopustnega. Takahama in Sakai v [217] sta izpostavila, da je za izboljšanje optimizacije problemov z enakostnimi omejitvami koristno nadzorovati interval ϵ, tako da prilagajamo njegov nivo vpliva. Algoritem ϵ-jDE uporablja metodo krmiljenje nivoja ϵ, tako da vektorje s kršitvami omejitev do nivoja ϵ obravnavamo enakovredno dopustnim vektorjem in primerjamo le njihove kriterijske ocene. Nivo ϵ prilagajamo, dokler število generacij g ne doseže nadzorne generacije G_{c}. Ko število

generacij preseže G_c, nivo ϵ postavimo na 0, da najdemo le vektorje z najmanjšo še dovoljeno kršitvijo omejitev. Metoda je definirana kot:

$$\epsilon_0 = \epsilon \tag{2.28}$$

$$v_0 = \overline{v}(\mathbf{x}_\theta) \tag{2.29}$$

$$\overline{v}_g = \begin{cases} \alpha_1 v_{g-1}, & \alpha_2 \overline{v}(\mathbf{x}_\beta) < v_{g-1},\ 0 < g < G_c \\ v_{g-1}, & \text{sicer} \end{cases} \tag{2.30}$$

$$\epsilon_G = \begin{cases} \max\{v_g(1-\frac{g}{G_c})^{c_p}, \epsilon\}, & 0 < g < G_c \\ 0, & \text{sicer} \end{cases}, \tag{2.31}$$

kjer predstavlja $\mathbf{x}_\theta$ najboljših θ vektorjev in $\theta = 0{,}3NP$. $\mathbf{x}_\beta$ označuje najboljših β vek-

Algoritem 2 Algoritem jDE [23] s strategijo 'rand/1/bin'.

Vhod: $\mathbf{f}(\mathbf{x})$ – kriterijska funkcija; D, NP, G – krmilni parametri DE

Izhod: $\mathbf{x}_{\text{best}}$ – optimirani parametri za dano kriterijsko funkcijo

1: `Uniformno naključno generiraj začetno populacijo DE` $\mathbf{x}_{i,0}$ `za` $i = 1..NP$;
2: **for** DE generacijska zanka g (dokler $g < G$) **do**
3: **for** DE iteracijska zanka i (za vse vektorje $\mathbf{x}_{i,g}$ v trenutni populaciji) **do**
4: DE izračun poskusnega vektorja $\mathbf{x}_{i,g}$ (samoprilagajanje, mutacija, križanje):
5: $$F_{i,g+1} = \begin{cases} F_l + rand_1 \times F_u & \text{če } rand_2 < \tau_1, \\ F_{i,g} & \text{sicer} \end{cases};$$
6: $$CR_{i,g+1} = \begin{cases} rand_3 & \text{če } rand_4 < \tau_2, \\ CR_{i,g} & \text{sicer} \end{cases};$$
7: $$\mathbf{v}_{i,g+1} = \mathbf{x}_{r_1,g} + F_{i,g+1}(\mathbf{x}_{r_2,g} - \mathbf{x}_{r_3,g});$$
8: $$u_{i,j,G+1} = \begin{cases} v_{i,j,g+1} & \text{če } rand(0,1) \leq CR_{i,g+1} \text{ ali } j = j_{\text{rand}} \\ x_{i,j,g} & \text{sicer} \end{cases};$$
9: DE selekcija z oceno ustreznosti $f(\mathbf{u}_{i,g+1})$:
10: $$\mathbf{x}_{i,G+1} = \begin{cases} \mathbf{u}_{i,g+1} & \text{če } f(\mathbf{u}_{i,g+1}) < f(\mathbf{x}_{i,g}) \\ \mathbf{x}_{i,g} & \text{sicer} \end{cases};$$
11: **end for**
12: **end for**
13: **vrni** najboljši najden posameznik ($\mathbf{x}_{\text{best}}$);

torjev in $\beta = 0{,}7NP$. Pri izračunu $\overline{v}(\mathbf{x}_\theta)$ privzamemo $\epsilon(0) = 0$. c_p je krmilni parameter hitrosti rahljanja dopustnosti omejitev in $\alpha_1 < 1$ ter $\alpha_2 > 1$ krmilita prilagajanje vrednosti v_g, ki prav tako krmili hitrost rahlanja dopustnosti omejitev. Parametra α_1 in α_2 lahko le zmanjšata vrednost v_g za majhno vrednost, ko je povprečna kršitev omejitev

$\overline{v}(\mathbf{x}_\beta)$ za najboljših β vektorjev, pomnožena z α_1, manjša od v_g. S tem prilagajanjem lahko nivo ϵ doseže vrednost 0 že pred $g \geq G_c$.

Algoritem jDE pozna še številne druge razširitve evolucijskih operatorjev, kot so dela [22, 20, 31, 25, 28, 21, 29, 33, 24, 32]. Algoritem jDE [23, 29] smo nadgradili tudi z večkriterijskim optimiranjem [30, 261], kar je obširneje razloženo v nadaljevanju. DE smo s soavtorji pomagali testirati tudi na dinamičnih problemih [120], kjer je algoritem zmagal na svetovnem tekmovanju. Za ta algoritem smo opravili še primerjavo z algoritmom DASA v revijalni objavi [24].

Zaradi svoje uspešnosti je algoritem DE bil že večkrat uporabljen tudi za večkriterijsko optimizacijo [2, 1, 145, 6, 7, 101, 35, 201, 227, 260, 205, 218, 3, 280, 265, 264, 126]. Na začetku raziskav na področju večkriterijske optimizacije so bile v uporabi klasične enokriterijske optimizacijske metode (npr. Newtonova metoda), pri katerih večkriterijski problem z utežno funkcijo pretvorimo v enokriterijski problem. Prvi poskus reševanja večkriterijskega problema z iskanjem več nedominiranih rešitev je bil predstavljen v [206]. Od takrat naprej [44, 45] se je nato pri večkriterijski optimizaciji precej povečal interes za uporabo populacijskih naključnih optimizacijskih algoritmov [58, 59, 284, 2, 285, 101, 201, 200, 63], kot so evolucijski algoritmi, simulirano ohlajanje [115], iskanje s tabuji [82] in drugi že omenjeni. S temi želimo najti čim več in čim ustrezneje porazdeljenih Pareto optimalnih rešitev. Vsi evolucijski algoritmi so populacijsko naravnani, saj ta pristop pride prav pri večkriterijski optimizaciji, kjer želimo odkriti množico rešitev, ki ponujajo številne kompromise med kriteriji. Pristop izdelave populacijskih vektorjev z diferencialno evolucijo so izbrali tudi že algoritmi za večkriterijsko optimizacijo, izmed katerih so prvi znani algoritmi [2, 6, 101, 201, 125]. Omenimo nekaj primerkov posameznih evolucijskih algoritmov:

NSGA-II (*Elitist Non-Dominated Sorting Genetic Algorithm*) je algoritem Deba in sodelavcev iz leta 2000 [60] in je zelo uspešen genetski algoritem (algoritem je nadgradnja NSGA [212]), ki se je v svojem času odrezal najbolje od vseh (takrat obstoječih) evolucijskih algoritmov za večkriterijsko optimizacijo. Algoritem uporablja **nedominirano urejanje** in **metriko nakopičenosti**, da tako izloča manj obetavne rešitve. Pri prvem gre za urejanje rešitev po frontah, pri čemer po vrsti za vsako velja, da nobena rešitev iz boljše fronte ni dominirana od rešitve iz naslednje slabše fronte. Metrika nakopičenosti pomaga razrešiti neodločene izide po prvem pristopu, t.j. čim bolj oddaljen od rešitev v isti fronti.

SPEA2 (*Strength Pareto Evolutionary Algorithm*) je algoritem Zitzlerja in sod. iz leta 2001 [286, 284] in je prav tako popularen genetski algoritem za večkriterijsko optimizacijo, ki glede na NSGA-II najde enakomerneje porazdeljene (manj nakopičene) rešitve. Vrednotenje posameznikov ne poteka po frontah, pač pa enovito z merjenjem **moči** (*strength*), **grobe uspešnosti** (*raw fitness*) in **gostote** (*density*) posameznikov. Moč posameznika je enaka številu posameznikov iz populacije, ki jih izbrani posameznik dominira. Kot zanimivost povejmo, da Zitzler in sod. v kalkulusu označujejo relacijo dominiranosti v obratni smeri, s simbolom $\succ$. Grobo spešnost posameznika izračunamo kot vsoto moči vseh posameznikov iz populacije, ki ta posameznik dominirajo. Ker z evolucijo številni posamezniki postanejo nedominirani (in imajo zato grobo uspešnost enako nič), potrebujemo dodatno informacijo o nakopičenosti rešitev. Zato za vse posameznike z enako grobo uspešnostjo izračunamo še gostoto. Algoritem za njen izračun je prilagojena različica algoritma k-tega najbližjega soseda [210]. Metriko nakopičenosti posameznika dobimo tako, da seštejemo grobo uspešnost in gostoto.

PDE (*Pareto-frontier Differential Evolution*) kot prvi predstavnik algoritmov diferencialne evolutije za večkriterijsko optimizacijo [2, 1, 145], kjer je v algoritmu dopolnjen selekcijski operator, tako da upošteva večkriterijsko selekcijo,

POGA (*Preference Ordering Genetic Algorithm*) [63] je primer nadgradnje večkriterijske selekcije algoritma NSGA-II, ki ima spremenjeno urejanje po frontah tako, da vpelje delno dominantnost, to je dominantnost na podmnožici kriterijev. V primeru, ko sta dva posameznika neprimerljiva oz. nedominantna, se ohrani posameznik, ki po čim več kriterijih dominira druge,

MTS (*Multiple Trajectory Search*) [224], ki z več trajektorijami neodvisno preiskuje kriterijski prostor,

MOPSO (Multiobjective Particle Swarm Optimization) [283], ki je razširitev algoritma optimizacije z rojem delcev na večkriterijsko optimizacijo,

MOEA/D (*Multiobjective Evolutionary Algorithm based on Decomposition*) [276], ki dekompozira večkriterijsko funkcijo in dosega dobre rezultate, vidne na primerjavi rezultatov iz tekmovanja CEC 2009 [278],

DEMO (*Differential Evolution for Multi-objective Optimization*), ki je ena prvih razširitev algoritma DE na večkriterijko optimizacijo [201],

GDE (*Generalized Differential Evolution*) [126], ki je podoben algoritmu DEMO, a ima drugače nastavljene nespremenljive vrednosti krmilnih parametrov,

MO-SADE (*Multi-objective Optimization based on Self-adaptive Differential Evolution*) [100], ki je razširitev enokriterijskega algoritma SADE [188] in daje povprečne rezultate [98],

MOjDE, DEMOwSA, DECMOSA-SQP naši algoritmi MOjDE (*Multi-objective Self-adaptive Differential Evolution*) [261], DEMOwSA (*Differential Evolution for Multi-objective Optimization with Self-adaptation*) [260], DECMOSA-SQP (*Differential evolution with self-adaptation and local search for constrained multiobjective optimization*) [265], ki so opisani v nadaljevanju in

stotine drugih večkriterijskih algoritmov za optimizacijio in njihovih izpeljank, ki so povečini bile predstavljene v revijah:

- IEEE Transactions on Evolutionary Computation (IEEE TEVC),
- Evolutionary Computation (EC),
- Soft Computing (SC),
- Applied Soft Computing (ASOC) in
- Information Sciences (IS)

ter na znanstvenih konferencah:

- IEEE Congress on Evolutionary Computation (CEC),
- IEEE World Congress on Computational Intelligence (WCCI) in
- Genetic and Evolutionary Computation Conference (GECCO).

Za enega od novejših pregledov večkriterijskih algoritmov za optimizacijo z literaturo v obsegu 310 člankov usmerimo bralca na aktualno objavo avtorjev Zhou s sod. [279] v novejši reviji Swarm and Evolutionary Computation (SWEVO).

Izboljšave diferencialne evolucije zajemajo tudi robustnost pri velikih dimenzijah iskalnega prostora [247, 263]. Križanje algoritma DE in evolucijskega programiranja [248] zasledimo v delu [246]. Algoritmu DE je soroden tudi algoritem optimizacije s kolonijami mravelj [120], zato smo izvedli tudi primerjalno študijo optimizacijskih mehanizmov in kakovosti optimiranja med algoritmoma [24]. Bošković s sodelavci smo

diferencialno evolucijo uporabljali za uglaševanje parametrov iger s popolno informacijo in ničelno vsoto [18, 17]. Algoritem DE je prav tako rotacijsko invarianten v prostoru spremenljivk [183]. Posledica tega je, da lahko dobro rešuje probleme z neločljivimi iskalnimi parametri. Takšno lastnost izkazuje precej realnih oz. industrijskih problemov [183, 74, 111, 223, 228, 282, 151, 238, 165]. Nad nekaterimi realnimi problemi iz dejanskega sveta smo preskusili tudi algoritem jDE [267, 256]. Tudi uporabnost večkriterijskih optimizacijskih algoritmov je bila prikazana že v različnih domenah [58]. Evolucijski algoritmi so se splošneje pokazali kot zelo uporabni pri reševanju kompleksnih večkriterijskih optimizacijskih problemov, vključno s številnimi problemi iz dejanskega sveta [267, 256]. Nekateri novejši evolucijski algoritmi za večkriterijsko optimizacijo rešujejo tudi probleme šuma v vhodnih podatkih [83]. Laumanns in sodelavci so enokriterijsko optimizacijo z omejitvami reševali z večkriterijsko optimizacijo [133]. Tan in sodelavci [218] so v večkriterijskem optimizacijskem algoritmu uporabili koevolucijo [42] in optimizacijo izvajali paralelno. Primer uporabe diferencialne evolucije za večkriterijsko optimizacijo realnih problemov vključuje tudi podatkovno rudarjenje [3]. Primeri večkriterijskih optimizacijskih problemov iz dejanskega sveta vključujejo še optimizacijo stacionarne plinske gorilne turbine [35], oblikovanje drobilca kamnov [10], problem distribucije naftnih derivatov [55], upravljanje z jedrskim gorivom [72], planiranje [209], načrtovanje telekomunikacijskih omrežij [236], obrambne aplikacije [102] in načrtovanje elektromotorjev [228].

Za oceno kakovosti optimizacijskih algoritmov so bile predlagane že številne metrike [118, 89], zato je večina obstoječih študij in primerjav algoritmov zasnovanih na specifičnih testnih meritvah. Te študije so največkrat statistične, izvedene nad izbranimi testnimi problemi, ki jih algoritem skuša rešiti. V nadaljevanju podajamo svoje nove algoritme za diferencialno evolucijo, njihova ocena zmogljivosti je tabelirana v dodatku A.

2.1.4 Naše dopolnitve diferencialne evolucije

V procesu priprave disertacije smo svetovni javnosti predstavili tudi nekaj novih algoritmov diferencialne evolucije, ki so prav tako vključeni v aplikativno domeno teze. Nekatere izmed njih smo do sedaj že omenili, kot so algoritmi SA-DE, DEwSA-cc, DEMOwSA, DECMOSA-SQP, MOjDE in $\text{jDE}_{\text{NP,MM}}$. Zaradi lažjega razumevanja sorodnih del in ozadja v tej disertaciji na kratko podajmo glavne značilnosti naših optimizacijskih algoritmov.

Algoritem SA-DE – diferencialna evolucija z logaritmično normalnim samoprilagajanjem krmilnih parametrov F in CR

Z algoritmom SA-DE [31] smo prvič v diferencialni evoluciji predstavili mehanizem normalnega prilagajanja samoprilagodljivih krmilnih parametrov F in CR in ga vstavili v originalni algoritem DE [215]. V procesu izdelave primerkov izdelamo nov primerek iz starševskih primerkov s pomočjo prilagoditve parametrov v operatorjih mutacije in križanja po vzoru iz evolucijskih strategij in nato nad njimi skupaj z vsakim posameznikom v operatorju selekcije izvedemo samoprilagajanje po vzoru jDE [23].

Mutacijo nad i-tim posameznikom $\mathbf{v}_{i,g+1}$ za generacijo $g+1$ v algoritmu SA-DE izvedemo s prilagoditvijo iz strategije rand/1/bin:

$$\mathbf{v}_{i,g+1} = \mathbf{x}_{r_1,g} + F_{i,g+1} \times (\mathbf{x}_{r_2,g} - \mathbf{x}_{r_3,g}), \tag{2.32}$$

kjer so $\mathbf{x}_{r_1,g}$, $\mathbf{x}_{r_2,g}$ in $\mathbf{x}_{r_3,g}$ vektorji uniformno naključno izbranih staršev iz generacije g. Nov faktor ojačanja diferenčnega vektorja F_i za vsakega od posameznikov i iz generacije g, za nov poskusni primerek v generaciji $g+1$ pri tem pred izdelavo posameznika prilagodimo:

$$F_{i,g+1} = \langle F_g \rangle_i \times \exp(\tau N(0,1)), \tag{2.33}$$

kjer τ označuje stopnjo učenja in je ponavadi sorazmeren s $\tau \sim 1/(\sqrt{2D})$ in D označuje dimenzijo problema. $N(0,1)$ je po Gaussu porazdeljena naključna spremenljivka. Izraz $\langle F_g \rangle_i$ označuje povprečenje parametra F i-tega obravnavanega posameznika in r_1, r_2 ter r_3 označujejo uniformno naključno izbrane posameznike iz generacije g:

$$\langle F_g \rangle_i = \frac{F_{i,g} + F_{r_1,g} + F_{r_2,g} + F_{r_3,g}}{4}, \tag{2.34}$$

kjer so indeksi r_1, r_2 in r_3 paroma ter od i različni indeksi posameznikov, ki so uporabljeni za izračun diference v mutacijskem procesu algoritma DE.

Tudi proces križanja v algoritmu SA-DE je povzet po vzoru iz strategije rand/1/bin, tako da z binarnim operatorjem križanja izdelamo poskusni vektor $u_{i,j,g+1}$:

$$u_{i,j,g+1} = \begin{cases} v_{i,j,g+1} & \text{if rand}(0,1) \leq CR_{i,g+1} \text{ ali } j = j_{\text{rand}} \\ x_{i,j,g} & \text{sicer,} \end{cases} \tag{2.35}$$

kjer $j \in \{1, ..., D\}$ označuje j-to komponento vektorja, $\text{rand}(0,1) \in [0,1]$ pomeni vzorčenje uniformno (psevdo) naključno porazdeljenega naključnega števila in j_{rand} pomeni uniformno naključen indeks iskalnega parametra, ki ga vedno izmenjamo (da bi s tem preprečili izdelavo enakih posameznikov). Tudi krmilni parameter CR smo po vzoru jDE [23] prilagajali in dobili $CR_{i,g+1}$ na drugačen način:

$$CR_{i,g+1} = \langle CR_g \rangle_i \times \exp(\tau N(0,1)), \tag{2.36}$$

kjer je τ podoben kot v prilagajanju parametra F in izraz $\langle CR_g \rangle_i$ ponovno označuje povprečenje:

$$\langle CR_g \rangle_i = \frac{CR_{i,g} + CR_{r_1,g} + CR_{r_2,g} + CR_{r_3,g}}{4}. \tag{2.37}$$

Študijo delovanja in primerjavo kakovosti algoritma SA-DE smo objavili skupaj s študijo delovanja algoritma jDE v poglavju v knjigi z zbirko nedavnega napredka na področju algoritma DE [31].

Algoritem DEwSAcc – diferencialna evolucija s samoprilagajanjem krmilnih parametrov in koevolucijo za probleme z velikimi dimenzijami

Algoritem DEwSAcc [263] je razširitev algoritma SA-DE [31] s še z dvema dodatnima mehanizmoma, tj. v algoritem DE [215] ob svojem normalnem načinu prilagajanja krmilnih parametrov vključimo še dve razširitvi. Prva razširitev je stohastična izbira med več mutacijskimi strategijami in druga je uvedba koevolucije.

V algoritmu DEwSAcc je mutacijski operator sestavljajen iz nabora treh mutacijskih strategij, iz katerega algoritem izbira naključno. V nabor smo vključili strategiji rand/1 in best/1 iz osnovnega DE [215] ter še strategijo rand/1/either-or [183]. Verjetnosti izbire teh strategij ob izdelavi mutacijskega vektorja so po vrsti 0,5, 0,1 in 0,4. Tako mutacijo za i-ti vektor $\mathbf{v}_{i,g+1}$ v novi generaciji $g+1$ definiramo kot:

$$\mathbf{v}_{i,g+1} = \begin{cases} \mathbf{x}_{r_1,g} + F_{i,g+1} \times (\mathbf{x}_{r_2,g} - \mathbf{x}_{r_3,g}) & \text{če } r_s < 0{,}5, \\ \mathbf{x}_{\text{best},g} + F_{i,g+1} \times (\mathbf{x}_{r_1,g} - \mathbf{x}_{r_2,g}) & \text{sicer če } r_s < 0{,}6, \\ \mathbf{x}_{r_1,g} + 0{,}5(F_{i,g+1} + 1) \times (\mathbf{x}_{r_2,g} + \mathbf{x}_{r_3,g} - 2\mathbf{x}_{r_1,g}) & \text{sicer,} \end{cases} \tag{2.38}$$

kjer so $\mathbf{x}_{r_1,g}$, $\mathbf{x}_{r_2,g}$ in $\mathbf{x}_{r_3,g}$ uniformno naključno izbrani starševski vektorji spremenljivk iz stare generacije g. Z $r_s \in [0,1]$ označimo uniformno naključno generirano število, ki izbira izmed tremi omenjenimi strategijami.

Učenje s koevolucijo so preučevali Chong s sod. [42]. Sodelovanje (ang. *cooperation*) med koevoluiranimi komponentami so preučevali Yang s sod. z algoritmom DECC2 [247], ki so na ta način reducirali dimenzije podfunkcij optimirane funkcije. Pokazali so tudi, da je pri optimiranju funkcij z velikim številom dimenzij učinkovito uporabiti koevolucijo po naslednjem načelu deli in vladaj:

1. dekompozicija problema – ločitev kriterijskih vektorjev v več manjših podkomponent,
2. optimiranje podkomponent – ločena evolucija podkomponent z različnimi optimizatorji in
3. združitev s sodelovanjem – ponovna združitev podkomponent v enovit sistem.

Algoritem DECC2 pri koevoluciji s parametrom cc_j določa, ali naj se j-to komponento funkcije vključi pri dekompoziciji:

$$cc_j = \begin{cases} 1 & \text{če } rand(0,1) \leq d_{i,g} \\ 0 & \text{sicer,} \end{cases} \tag{2.39}$$

kjer je $d_{i,g}$ dekompozicijska stopnja posameznega vektorja, prilagajana skozi generacije:

$$d_{i,g+1} = d_{i,g} \times \exp(\tau N(0,1)). \tag{2.40}$$

Opisan način koevolucije je možno uporabiti samo nad ločljivo kriterijsko funkcijo, npr. večkriterijsko funkcijo. Pri optimiranju neločljivih funkcij tega pristopa ne moremo izbrati, zato smo sami pripravili osnutek mehanizma za koevolucijo neločljivih funkcij in ga predstavili z algoritmom DEwSAcc [263]. V koevolucijskem mehanizmu DEwSAcc dimenzije vektorja spremenljivk razstavimo na komponente in jih nekaj generacij optimiramo ločeno. Ne razstavimo torej kriterijskega vektorja, temveč vektor spremenljivk. Ta vektor razstavimo na komponente, ki jih spreminjamo in tiste, ki jih pustimo nespremenjene. Ločitev izvedemo tako, da nabor komponent vektorja spremenljivk, ki jih v poskusnem vektorju dovolimo spreminjati, fiksiramo za določeno prilagodljivo število generacij $G^{\text{cc}}_{i,g}$:

$$G^{\text{cc}}_{i,g+1} = G^{\text{cc}}_{i,g} \times \exp(\tau N(0,1)). \tag{2.41}$$

V algoritmu DEwSAcc koevolucijo tako izvedemo za poskusni vektor $u_{i,j,g+1}$ z binarnim operatorjem križanja:

$$u_{i,j,g+1} = \begin{cases} v_{i,j,g+1} & \text{če } cc_j = 1 \text{ in } (rand(0,1) \leq CR_{i,g+1} \text{ ali } j = j_{\text{rand}}) \\ x_{i,j,g} & \text{sicer.} \end{cases} \tag{2.42}$$

Algoritem DEMOwSA – diferencialna evolucija za večkriterijsko optimizacijo z logaritmično normalnim samoprilagajanjem

Po izboljšavi osnovnega algoritma DE s svojevrstnim logaritmično normalnim samoprilagajanjem krmilnih parametrov iz algoritma SA-DE smo razširitev uvedli še v večkriterijsko optimizacijo, v algoritem DEMO. Tako smo mehanizem samoprilagajanja iz algoritma SA-DE (enokriterijska optimizacija) vključili v novem algoritmu DEMOwSA [260, 255] za večkriterijsko optimizacijo, ki temelji na algoritmu DEMO [201]. Glavna cilja te vključitve sta bila (1) predstaviti izboljšan algoritem za večkriterijsko optimizacijo, ki v povprečju dosega boljše rezultate, kot jih dosegajo nekateri obstoječi algoritmi in (2) vgraditi samoprilagajanje nekaterih krmilnih parametrov, tako da jih uporabniku ni potrebno več nastavljati, in s tem povečati robustnost algoritma za kasnejšo uporabo na novih optimizacijskih problemih.

$x_{1,1,G}$	$x_{1,2,G}$	$x_{1,3,G}$	...	$x_{1,D,G}$	$F_{1,G}$	$CR_{1,G}$	$f_1(\mathbf{x}_{1,G})$	$f_2(\mathbf{x}_{1,G})$	$f_3(\mathbf{x}_{1,G})$	...	$f_M(\mathbf{x}_{1,G})$
$x_{2,1,G}$	$x_{2,2,G}$	$x_{2,3,G}$	...	$x_{2,D,G}$	$F_{2,G}$	$CR_{2,G}$	$f_1(\mathbf{x}_{2,G})$	$f_2(\mathbf{x}_{2,G})$	$f_3(\mathbf{x}_{2,G})$	...	$f_M(\mathbf{x}_{2,G})$
$x_{3,1,G}$	$x_{3,2,G}$	$x_{3,3,G}$	...	$x_{3,D,G}$	$F_{3,G}$	$CR_{3,G}$	$f_1(\mathbf{x}_{3,G})$	$f_2(\mathbf{x}_{3,G})$	$f_3(\mathbf{x}_{3,G})$	...	$f_M(\mathbf{x}_{3,G})$
⋮	⋮	⋮	⋱	⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮
$x_{NP,1,G}$	$x_{NP,2,G}$	$x_{NP,2,G}$	...	$x_{NP,D,G}$	$F_{NP,G}$	$CR_{NP,G}$	$f_1(\mathbf{x}_{NP,G})$	$f_2(\mathbf{x}_{NP,G})$	$f_3(\mathbf{x}_{NP,G})$	...	$f_M(\mathbf{x}_{NP,G})$

Slika 2.2: Predstavitev samoprilagodljivih krmilnih parametrov in ovrednotenja v posamezniku $\mathbf{p} = \{\mathbf{x}, \mathbf{s}, \mathbf{f}(\mathbf{x})\}$.

Algoritem 3 Ovrednoti$_{OKA2}$: primer večkriterijske funkcije OKA2 [98].

Vhod: $\mathbf{p}$ – primerek posameznega vektorja $\mathbf{p} = \{\mathbf{x}, \mathbf{s}, \mathbf{f}(\mathbf{x})\}$ funkcije $\mathbf{f}(\mathbf{x})$.

Izhod: Izračunana $\mathbf{f}(\mathbf{x})$, ki se shrani v $\mathbf{p}$ (za ponazoritev predstavitve primerka glej sliko 2.2).

1: $x_1, x_2, x_3 \leftarrow \mathbf{p}$ {Iz $\mathbf{p}$ izlušči iskalne parametre, $x_1 \in [-\pi, \pi], x_2, x_3 \in [-5, 5]$.}
2: $f_1(\mathbf{x}) = x_1$
3: $f_2(\mathbf{x}) = 1 - \frac{1}{4\pi^2}(x_1 + \pi)^2 + |x_2 - 5\cos(x_1)|^{\frac{1}{3}} + |x_3 - 5\sin(x_1)|^{\frac{1}{3}}$
4: $\mathbf{f}(\mathbf{x}) = (f_1, f_2)$
5: $\mathbf{p} \leftarrow \mathbf{f}(\mathbf{x})$ {V $\mathbf{p}$ shrani vrednosti kriterijev.}

Izvorni kod in idejna zasnova algoritma DEMOwSA sloni na izvornem kodu osnovnega algoritma DEMO Tušarjeve in Filipiča [201, 229]. Algoritem DEMO je algoritem

diferencialne evolucije [215, 214] za večkriterijsko optimizacijo, ki uporablja fiksne krmilne parametre diferencialne evolucije ($F = 0{,}5$, $CR = 0{,}3$). Omogoča izbiro več selekcijskih mehanizmov NSGA-II [59], SPEA2 [284] ali IBEA [285], ki jih izberemo pred optimizacijskim zagonom. Izboljšane vektorje shranjuje v arhivsko aproksimacijsko množico, ki predstavlja ob enem tudi trenutno populacijo. Glavna razlika v algoritmu DEMOwSA od algoritma DEMO je v uporabi operatorja mutacije. Algoritmu DEMO [201, 227, 229] smo dodali zmožnost samoprilagajanja krmilnih parametrov za izdelavo poskusnega vektorja po vzoru evolucijskih strategij, z logaritmično normalno porazdeljenim prilagajanjem. Vsak posameznik (glej sliko 2.2) algoritma DEMOwSA je razširjen s samoprilagodljivimi krmilnimi parametri F in CR. Vrednosti teh parametrov se samoprilagajajo skozi proces iskanja, saj so krmilni parametri predstavljeni v vsakem posamezniku poleg ostalih parametrov. Izbrali smo še SPEA2 večkriterijsko selekcijo in ohranili ostale nastavitve parametrov v algoritmu DEMO. Da bi razumeli celoten algoritem DEMOwSA, nam bo lažje, če ga osvojimo po delih. Zato bomo najprej predstavili nekaj enostavnejših netrivialnih podalgoritmov, nato pa podali celovito sliko algoritma v psevdokodu, ki združuje zapisane manjše kose celega algoritma.

Postopek za ovrednotenje posameznih primerkov je lahko preprost. Ponavadi izvede funkcijo, takšen primer podaja tudi algoritem Ovrednoti$_{OKA2}$ (glej algoritem 3).

En od preprostih manjših postopkov je tudi primerjava kriterijskih vektorjev, ki ga ponazarja algoritem Primerjaj posameznika (glej algoritem 4), njegovo časovno zahtevnost glede na DEMO smo nekoliko izboljšali, kar je označeno v vrstici 11. Ta medsebojno primerja dve rešitvi in vrne, katera je boljša ali pa razsodi, da sta rešitvi neprimerljivi, t.j. nobena ni boljša. Pri primerjavi se upošteva pojem dominiranja, definiran v uvodu.

Komentirajmo algoritem Premešaj populacijo (glej algoritem 5, ter sliko 2.3 za primer takšnega naključnega premešanja), ki ga uporabimo po vsaki generaciji. Ta deluje tako, da seznam $\mathbf{a} = (12, 18, 3, -5, 7)$ naključno premeša (a). Zato za vsak element seznama zgenerira enakomerno porazdeljeno naključno število med 0 in 1 (b) in si ob tem zabeleži še pripadajočo številko indeksa, h kateremu elementu število spada, (c). Seznam uredi po generiranih naključnih vrednostih (d), nato začeten seznam $\mathbf{a}$ sprazni (e) in ga zaporedoma znova napolni (f–h), tako da po vrsti vstavlja elemente iz dobljenih indeksov v koraku (e).

Populacijske vektorje algoritem DEMOwSA inicializira, kot je prikazano v algoritmu Inicializiraj nov posameznik (glej algoritem 6). Ustvarjanje novih vektorjev in s tem

Algoritem 4 Primerjaj posameznika.

Vhod: $\mathbf{p}_i$ – starš iz trenutne populacije; $\mathbf{u}$ – nov posameznik; M – dimenzija prostora kriterijev izbrane funkcije.

Izhod: ND – nedominiranost (1/0 – nov/starš dominanten, 0 – neprimerljiva).

1: $d_{\text{new}} = 0$ {Zastavica, ali je nov posameznik dominiran po enem od kriterijev.}
2: $d_{\text{parent}} = 0$ {Zastavica, ali je starš dominiran po enem od kriterijev.}
3: **for** $j = 0$ to M **do**
4: **if** $\mathbf{u}_j < \mathbf{p}_{i,j}$ **then**
5: $d_{\text{parent}} = 1$
6: **else if** $\mathbf{p}_{i,j} < \mathbf{u}_j$ **then**
7: $d_{\text{new}} = 1$
8: **end if**
9: **if** $d_{\text{parent}} = 1$ **and** $d_{\text{new}} = 1$ **then**
10: $ND := 0$ {Posameznika sta neprimerljiva, noben ne dominira drugega.}
11: **return** {Takoj prenehamo s primerjavo — izboljšava v DEMOwSA.}
12: **end if**
13: **end for**
14: $ND := d_{\text{new}} - d_{\text{parent}}$ {1 – nov je dominanten; -1, starševski je dominanten.}

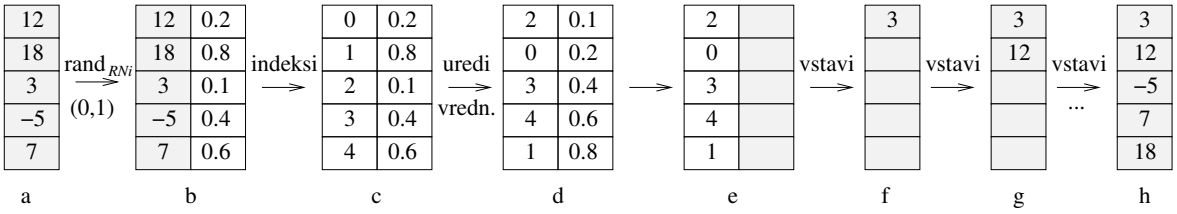
Algoritem 5 Premešaj populacijo.

Vhod: $\mathbf{P}_g$ – vhodna populacija primerkov v generaciji g, pred pričetkom izdelave novih primerkov; NP – velikost populacije in aproksimacijske množice $\overline{N}$; RNi – trenutno stanje generatorja enakomerno porazdeljenih naključnih števil.

Izhod: $\mathbf{P}'_g$ – delovna populacija primerkov za generacijo g, med tvorjenjem novih primerkov.

Izhod: Spremenjen RNi.

1: $\mathbf{r} := \emptyset$ {Seznam naključnih števil.}
2: $\mathbf{r}' := \emptyset$ {Seznam indeksov, glej sliko 2.3.}
3: **for** $i = 0$ to $NP - 1$ **do**
4: $\mathbf{r} := \mathbf{r} \cup \text{rand}_{RNi}(0, 1)$ {Dodajaj naključna števila – slika 2.3b.}
5: $\mathbf{r}' := \mathbf{r} \cup i$ {Pred vsak element seznama $\mathbf{r}$ shrani indeks i – slika 2.3c.}
6: **end for**
7: Seznam $\mathbf{rr}'$ uredi po $\mathbf{r}$. {Hitro urejanje: $\Theta(NP \log NP)$ – slika 2.3d.}
8: $\mathbf{P}'_g := \emptyset$ {Nov seznam primerkov v populaciji.}
9: **for** $i = 0$ **to** $NP - 1$ **do**
10: $\mathbf{P}'_g := \mathbf{P}'_g \cup \mathbf{p}_{g,r_i}$ {Iz indeksov r_i iz populacije $\mathbf{P}_g$ dodajaj v $\mathbf{P}'_g$ – slika 2.3e–h.}
11: **end for**



Slika 2.3: Prikaz algoritma za premešanje seznama na preprostem primeru.

Algoritem 6 Inicializiraj nov posameznik.

Vhod: $func\_type$ – zaporedna številka funkcije, ki jo optimiziramo; $\mathbf{g}(\mathbf{x}) \geq 0$ – omejitve optimizirane funkcije; D – dimenzija prostora spremenljivk; M – dimenzija prostora kriterijev; F_{init} – začetna vrednost parametrov F; CR_{init} – začetna vrednost parametrov CR; RNi – seme generatorja enakomerno porazdeljenih naključnih števil; RNj – seme generatorja normalno porazdeljenih naključnih števil.

Izhod: $\mathbf{p}$ – nov primerek rešitve večkriterijske funkcije in RNi – spremenjeno seme.

1: $\mathbf{a} = \emptyset$
2: **for** $j = 0$ to D **do**
3: $\quad a_j := \text{rand}_{RNi}(x_{\min,i}, x_{\max,i})$ $\{x_{\min,i}$ in $x_{\max,i}$ vsebovana v omejitvah $\mathbf{g}(\mathbf{x}) \geq 0.\}$
4: **end for**
5: $\mathbf{s} = (F_{\text{init}}, CR_{\text{init}})$
6: $\mathbf{G} = \mathbf{0}_{M+1}$ $\{M$ kriterijev in nedominantnost.$\}$
7: $\mathbf{p} = (\mathbf{a}, \mathbf{s}, \mathbf{G})$ {Sestavi primerek iz iskalnih parametrov in krmilnih parametrov.}

preiskovanje iskalnega prostora poteka v algoritmu Nov primerek (glej algoritem 7). Ta iz starša $\mathbf{p}_i$ in diferenc med vektorji v populaciji $\mathbf{P}_g$ iz generacije g ustvari nov poskusni vektor $\mathbf{u}$ za generacijo $g + 1$.

Ker je populacijskih vektorjev pred koncem vsake generacije lahko preveč, je nekatere potrebno izločiti, oz. populacijo t.i. porezati. Takšen postopek prikazuje algoritem Poreži$_{\text{SPEA2}}$ (glej algoritem 8), ki je natančneje opisan še z Nakopičenost$_{\text{SPEA2}}$ (glej algoritem 10) in Odstrani prenakopičene vektorje$_{\text{SPEA2}}$ (glej algoritem 9).

Sedaj imamo pripravljene vse sestavne elemente algoritma DEMOwSA. Zato lahko sedaj načrt celotnega algoritma v obliki psevdokoda podamo v algoritmu 11. Kot vidimo, algoritem 11 vsebuje prej omenjene manjše algoritme. Omenimo lahko še, da je algoritem DEMOwSA izboljšan v algoritmu 4 glede na algoritem DEMO. Spremenjeni so tudi ostali, razen algoritmov 5 in 8, 10 ter 9, ki so enaki kot v algoritmu DEMO oz. so iz algoritma SPEA2 [284]. Glavna prednost algoritma DEMOwSA glede na algoritem DEMO je v spremenjenem algoritmu Nov primerek (glej algoritem 7).

Algoritem 7 Nov primerek$_{\text{DEMOwSA/rand/1/bin}}$.

Vhod: $\mathbf{p}_i$ – starševski primerek rešitve; D – dimenzija prostora spremenljivk; τ – stopnja učenja za prilagajanje samoprilagodljivih parametrov F in CR; $F_{\min}$ ($F_{\max}$) – najmanjša (največja) možna vrednost parametra F; $CR_{\min}$ ($CR_{\max}$) – najmanjša (največja) možna vrednost parametra CR; $\mathbf{g}(\mathbf{x}) \geq 0$ – omejitve funkcije, ki jo optimiziramo; RNi – seme generatorja enakomerno porazdeljenih naključnih števil; RNj – seme generatorja normalno porazdeljenih naključnih števil; $\mathbf{P}_g$ – populacija primerkov v generaciji g.

Izhod: Poskusni vektor $\mathbf{u}$ za generacijo $g+1$ in spremenjena RNi ter RNj.

repeat
  $r_1 := \text{rand}_{RNi}(0, NP-1)$; $r_2 := \text{rand}_{RNi}(0, NP-1)$; $r_3 := \text{rand}_{RNi}(0, NP-1)$
until not ($i = r_1$ **or** $i = r_2$ **or** $i = r_3$ **or** $r_1 = r_3$ **or** $r_2 = r_3$) {Izberi paroma različne indekse $\mathbf{p}_{g,r_1}$, $\mathbf{p}_{g,r_2}$, $\mathbf{p}_{g,r_3} \in \mathcal{P}_g$.}
$\langle F_g \rangle_i = \dfrac{F_{i,g} + F_{r_1,g} + F_{r_2,g} + F_{r_3,g}}{4}$ {Globalno povprečenje skozi primerke.}
$F_{i,g+1} = \langle F_g \rangle_i \times \exp(\tau N(0,1))$ {Logaritmično normalno prilagajanje.}
if $F_{i,g+1} < F_{\min}$ **then**
  $F_{i,g+1} := F_{\min}$ {Popravimo F v meje.}
else if $F_{i,g+1} > F_{\max}$ **then**
  $F_{i,g+1} := F_{\max}$
end if
$\mathbf{v}_{i,g+1} = \mathbf{x}_{r_1,g} + F_{i,g+1} \times (\mathbf{x}_{r_2,g} - \mathbf{x}_{r_3,g})$
$\langle CR_g \rangle_i = \dfrac{CR_{i,g} + CR_{r_1,g} + CR_{r_2,g} + CR_{r_3,g}}{4}$ {Globalno povprečenje primerkov.}
$CR_{i,g+1} = \langle CR_g \rangle_i \times \exp(\tau N(0,1))$ {Logaritmično normalno prilagajanje.}
if $CR_{i,g+1} < CR_{\min}$ **then**
  $CR_{i,g+1} := CR_{\min}$ {Popravimo CR v meje.}
else if $CR_{i,g+1} > CR_{\max}$ **then**
  $CR_{i,g+1} := CR_{\max}$
end if
$j_{\text{rand};g,i} := \text{rand}_{RNi}(0, D)$ {Vsaj enega parametra pri križanju ne bomo zamenjali.}
for $j = 0$ **to** D **do**
  if $\text{rand}_{RNi}(0,1) \leq CR_{i,g+1}$ ali $j = j_{\text{rand};g,i}$ **then**
    $u_{i,j,g+1} = v_{i,j,g+1}$ {Križamo.}
  else
    $u_{i,j,g+1} = x_{i,j,g}$ {Ne križamo.}
  end if
  if $u_{i,j,g+1} < x_{\min,i}$ **then**
    $u_{i,j,g+1} := x_{\min,i}$ {$x_{\min,i}$ in $x_{\max,i}$ sta vsebovana v omejitvah $\mathbf{g}(\mathbf{x}) \geq 0$.}
  else if $u_{i,j,g+1} > x_{\max,i}$ **then**
    $u_{i,j,g+1} := x_{\max,i}$
  end if
end for

Algoritem 8 Poreži$_{\text{SPEA2}}$.

Vhod: NP – velikost populacije in aproksimacijske množice $\overline{N}$; $\mathbf{P}_g$ – populacija primerkov v generaciji g.

Izhod: $\mathbf{P}_{g+1}$ – na podlagi nakopičenosti do velikosti NP porezana populacija $\mathbf{P}_g$.

1: $F(\mathbf{P}_g), \sigma :=$ Nakopičenost$_{\text{SPEA2}}(\mathbf{P}_g)$ {Ocena nakopičenosti vsakega vektorja.}
2: $\mathbf{P}_{g+1} :=$ Odstrani prenakopičene vektorje$_{\text{SPEA2}}(NP, \mathbf{P}_g, F(\mathbf{P}_g), \sigma)$

Algoritem 9 Odstrani prenakopičene vektorje$_{\text{SPEA2}}$.

Vhod: NP – velikost populacije in aproksimacijske množice $\overline{N}$; $\mathbf{P}_g$ – populacija primerkov v generaciji g; $F(\mathbf{P}_g)$ in σ – matrika nakopičenosti in matrika razdalj do k-tega primerka.

Izhod: $\mathbf{P}_{g+1}$ – na podlagi nakopičenosti do velikosti NP porezana populacija $\mathbf{P}_g$.

1: $\mathbf{P}_{g+1} := \emptyset$ {Tu se prične kopiranje primerkov v novo generacijo.}
2: **for all** $\mathbf{p}_{g,i}$ **with** $F(\mathbf{p}_{g,i}) < 1$ **do**
3: $\quad\mathbf{P}_{g+1} := \mathbf{P}_{g+1} \cup \mathbf{p}_{g,i}$ {Prekopiraj prvo fronto: vse nedominirane iz $\mathbf{P}_g$ v $\mathbf{P}_{g+1}$.}
4: $\quad\mathbf{P}_g := \mathbf{P}_g \setminus \mathbf{p}_{g,i}$ {Vektorje prve fronte odstranjujemo, da bo $F(\mathbf{p}_{g,i}) \geq 1$.}
5: **end for**
6: **if** $|\mathbf{P}_{g+1}| < NP$ **then**
7: $\quad$Uredi $\mathbf{P}_{g+1}$ naraščajoče po $F(\mathbf{p}_{g,i})$ {Hitro urejanje: $\Theta(\mathbf{P}_{g+1} \log \mathbf{P}_{g+1})$.}
8: $\quad i := 0$ {Boljši (t.j. s čim manjšo nakopičenostjo $F(\mathbf{p}_{g,i})$) so po urejanju spredaj.}
9: $\quad$**repeat**
10: $\quad\quad\mathbf{P}_{g+1} := \mathbf{P}_{g+1} \cup \mathbf{p}_{g,i}$
11: $\quad\quad i := i + 1$ {Kopiramo v smeri od boljših k slabšim, do zapolnitve arhiva.}
12: $\quad$**until** $|\mathbf{P}_{g+1}| = NP$
13: **else if** $|\mathbf{P}_{g+1}| > NP$ **then**
14: $\quad$**repeat**
15: $\quad\quad k := 0$ {Poiščemo najmanj oddaljen primerek $\mathbf{p}_{g+1,i}$:}
16: $\quad\quad$**for all** $\mathbf{p}_{g+1,i} \in \mathbf{P}_{g+1}$ **do**
17: $\quad\quad\quad$**for all** $\mathbf{p}_{g+1,j} \in \mathbf{P}_{g+1}$ **do**
18: $\quad\quad\quad\quad$**if** $\forall 0 < k < |\mathbf{P}_{g+1}| : \sigma_i^k = \sigma_j^k \ \vee$
$\quad\quad\quad\quad\quad\exists 0 < k < |\mathbf{P}_{g+1}| : \left[\left(\forall 0 < l < k : \sigma_i^l = \sigma_j^l\right)\right] \wedge \sigma_i^k < \sigma_j^k$ **then**
19: $\quad\quad\quad\quad\quad\mathbf{P}_{g+1} := \mathbf{P}_{g+1} \setminus \mathbf{p}_{g+1,k}$ {Odstranimo najbolj nakopičenega, čas $\Theta(M)$.}
20: $\quad\quad\quad\quad\quad$**break for** i {Dokler velikost presega NP, še izvajaj operator sekanja.}
21: $\quad\quad\quad\quad$**end if**
22: $\quad\quad\quad$**end for**
23: $\quad\quad$**end for**
24: $\quad$**until** $|\mathbf{P}_{g+1}| = NP$
25: **end if**

Algoritem 10 Nakopičenost$_{\text{SPEA2}}$.

Vhod: NP – velikost populacije in aproksimacijske množice $\overline{N}$; $\mathbf{P}_g$ – populacija primerkov v generaciji g.

Izhod: $F(\mathbf{P}_g)$ in σ – matrika nakopičenosti in matrika razdalj do k-tega primerka.

$\mathbf{S}_g := \mathbf{0}$ {Primerkom iz $\mathbf{P}_g$ inicializiraj moč na 0 – še niso dominirani.}
for all $\mathbf{p}_{g+1,i} \in \mathbf{P}_{g+1}$ **do**
  for all $\mathbf{p}_{g+1,j} \in \mathbf{P}_{g+1}$ **do**
    if Primerjaj posameznika$(\mathbf{p}_{g,i}, \mathbf{p}_{g,j}, M) = 1$ **then**
      $S_{g,i} := S_{g,i} + 1$ {Prešteje dominirane $S_{\mathbf{p}_{g,i}} = |\{\mathbf{p}_{g,j} | \mathbf{p}_{g,j} \in \mathbf{P}_g\} \wedge \mathbf{p}_{g,i} \prec \mathbf{p}_{g,j}|$}
    end if
  end for
end for
$\mathbf{R}_g := \mathbf{0}$ {Grobo uspešnost izračunamo iz števila dominiranih vektorjev $\mathbf{p}_{g,i}$ v $\mathbf{S}_g$.}
for all $\mathbf{p}_{g+1,i} \in \mathbf{P}_{g+1}$ **do**
  for all $\mathbf{p}_{g+1,j} \in \mathbf{P}_{g+1}$ **do**
    if Primerjaj posameznika$(\mathbf{p}_{g,i}, \mathbf{p}_{g,j}, M) = 1$ **then**
      $R_{g,i} := R_{g,i} + R_{g,j}$ {Vsota moči dominiranih, manjša vrednost je boljša.}
    end if
  end for
end for
$\mathbf{n}_{\text{n}} = -\mathbf{1}_{|\mathbf{P}_{g+1}| \times |\mathbf{P}_{g+1}|}$ {Kvadratna matrika velikosti $|\mathbf{P}_{g+1}|$, čas $\Theta((|\mathbf{P}_{g+1}|)^2)$.}
$\mathbf{c} = \mathbf{1}_{|\mathbf{P}_{g+1}|}$ {Vektor enic velikosti $|\mathbf{P}_{g+1}|$.}
for all $\mathbf{p}_{g+1,i} \in \mathbf{P}_{g+1}$ **do**
  $n_{\text{n};i,0} := i$ {Zgradimo vektor razdalj za vsak primerek.}
  for $j := 0$ **to** i **do**
    $\sigma_{i,j} := \sigma_{j,i}$ {Že izračunane dolžine povezav samo prekopiramo.}
  end for
  $\sigma_{i,i} := 0$ {Razdalja do samega sebe je nič.}
  for $j := i + 1$ **to** $|\mathbf{P}_{g+1}|$ **do**
    $\sigma_{i,j} := \sum_{i=0}^{M} \left(f_i(\mathbf{x}_{g+1,i}) - f_i(\mathbf{x}_{g+1,j})\right)^2$ {Geometrijska razdalja, $\Theta(M)$.}
    if $\sigma_{i,j} = 0$ **then**
      $n_{\text{n};i,c_i} := j$ {Indeksi podvojenih vektorjev: na indeksu i, podvojen na c_i.}
      $c_i := c_i + 1$ {Štejemo število kopij enakih vektorjev.}
      $c_j := c_j + 1$ {Štejemo število kopij enakih vektorjev – tudi v nasprotni smeri.}
    end if
  end for
end for
for $i := 0$ **to** $|\mathbf{P}_{g+1}|$ **do**
  $F(\mathbf{p}_{g,i}) := \dfrac{R_{g,i} + 1}{\sigma_{i,k} + 2}$ {Nakopičenost: moč napram razdalji k-tega soseda.}
end for

Algoritem 11 Diferencialna evolucija za večkriterijsko optimizacijo s samoprilagajanjem.

Vhod: $func\_type$ – zaporedna številka funkcije, ki jo optimiziramo; $\mathbf{g}(\mathbf{x}) \geq 0$ – omejitve optimizirane funkcije; D – dimenzija prostora spremenljivk; M – dimenzija prostora kriterijev; *MAXFEs* – maksimalno število ovrednotenj (ustavitveni pogoj); NP – velikost populacije in aproksimacijske množice $\overline{N}$; s_{DE} – številka strategije mutacije DE (strategija preiskovanja); sel_type_{MO} – strategija večkriterijske selekcije (strategija sekanja); F_{init} – začetna vrednost parametrov F; CR_{init} – začetna vrednost parametrov CR; F_{min} (F_{max}) – najmanjša (največja) možna vrednost parametra F; CR_{min} (CR_{max}) – najmanjša (največja) možna vrednost parametra CR; τ – stopnja učenja za prilagajanje samoprilagodljivih parametrov F in CR; RNi_{seed} – seme generatorja enakomerno porazdeljenih naključnih števil; **front_gen** – množica intervalov vzorčenj populacije $\mathbf{P}_g$ po FE.

Izhod: Seznam aproksimacijskih množic iskalnih parametrov $\mathcal{P}$ (množica matrik).

$RNj_{\text{seed}} :=$ rand(RNi) {Seme generatorja normalno porazdeljenih števil RNj_{seed}.}
$g = 0$ {Z g štej generacije populacij od 0 naprej.}
for $i = 0$ **to** $NP - 1$ **do**
  $\mathbf{p}_{0,i} :=$ Inicializiraj nov primerek($func\_type, D, M, F_{\text{init}}, CR_{\text{init}}, RNi, RNj$)
  Ovrednoti$_{func_type}(\mathbf{p}_{0,i})$ {Ovrednoti primerek v začetni populaciji.}
  $\mathbf{P}_0 := \mathbf{P}_0 \cup \mathbf{p}_{0,i}$ {Dopolnjuj začetno populacijo $\mathbf{P}_0$.}
end for
FEs := *NP* {Ovrednotenih je *NP* primerkov.}
while *FEs* < *MAXFEs* **do**
  if $FEs \geq front\_gen_0$ **then**
    $\mathcal{P} := \mathcal{P} \cup \mathbf{P}_g^{NP,D}$ {V $\mathcal{P}$ dodaj D iskalnih parametrov za vseh *NP* primerkov.}
    front_gen := **front_gen** $\setminus\, front\_gen_0$ {Odstrani prvi element množice.}
  end if
  $\mathbf{P}_g :=$ Premešaj populacijo($\mathbf{P}_g, NP, RNi$) {Naključno oštevilči osebke iz $\mathbf{P}_g$.}
  for $i = 0$ **to** $NP - 1$ **do**
    $\mathbf{u} :=$ Nov primerek$_{s_{\text{DE}}}(\mathbf{p}_i, D, \tau, F_{\text{min}}, F_{\text{max}}, CR_{\text{min}}, CR_{\text{max}}, RNi, RNj, \mathbf{P}_g)$
    Ovrednoti$_{func_type}(\mathbf{u})$ {Ovrednoti primerek glede na funkcijo $func\_type$.}
    FEs := *FEs* + 1 {Beleži število ovrednotenj.}
    Primerjaj posameznika($\mathbf{p}_i, \mathbf{u}, M$) {Dominiranost v kriterijskem prostoru.}
    if $ND_u = 1$ **then**
      $\mathbf{P}_g := (\mathbf{P}_g \setminus \mathbf{P}_i) \cup \mathbf{u}$ {Primerek $\mathbf{u}$ dominanten, prepiši starša s primerkom.}
    else
      if $ND_u = -1$ **then**
        $\mathbf{P}_g := \mathbf{P}_g$ {Starš $\mathbf{p}_i$ dominanten, primerek $\mathbf{u}$ izbriši}.
      else
        $\mathbf{P}_g := \mathbf{P}_g \cup \mathbf{u}$ {Dodaj primerek $\mathbf{u}$ v aproksimacijsko množico.}
      end if
    end if
  end for
  $\mathbf{P}_{g+1} =$ Poreži$_{sel_type_{\text{MO}}}(\mathbf{P}_g, NP)$ {Poreži populacijo $\mathbf{P}_g$ do velikosti *NP*.}
  $g := g + 1$ {Povečaj števec generacij.}
end while

Algoritem DEMOwSA smo podrobno predstavili, analizirali in izmerili njegovo učinkovitost v magistrskem delu [255]. Ugotovili smo, da je za reševanje večkriterijskih optimizacijskih problemov naš algoritem po kakovosti dobljenih vektorjev primerljiv z najsodobnejšimi konkurenčnimi algoritmi in boljši od nekaterih obravnavanih. Izkazuje statistično signifikantne izboljšave kakovosti na številnih aktualnih testnih funkcijah. Rezultati algoritma, časovna analiza, ocena kakovosti algoritma z izračunom indikatorjev za testirane standardne funkcije in izris ploskev dosega in statistično primerjavo kakovosti z analizo dinamike samoprilagodljivih krmilnih parametrov, ki pokažeta prednosti in slabosti algoritma ter obnašanje mehanizma samoprilagajanja, so podrobno razdelani v magistrski nalogi [255] in jih tukaj ne ponavljamo. V omenjeni magistrski nalogi so v dodatkih zbrani še enovit psevdokod predstavljenega algoritma, uporabljane testne funkcije in opis paralelnega ogrodja za izvedbo eksperimentov.

Algoritem DEMOwSA-SQP – diferencialna evolucija za večkriterijsko optimizacijo z logaritmično normalnim samoprilagajanjem in lokalnim iskanjem

Predstavili smo tudi hibridni algoritem za večkriterijsko optimizacijo DEMOwSA-SQP [262]. Kombinirali smo samoprilagodljivo diferencialno evolucijo SA-DE [31] za globalno preiskovanje in sekvenčno kvadratično programiranje SQP [16] za lokalno preiskovanje. Ta način preiskovanja smo uporabili pri izdelavi novih poskusnih vektorjev v optimizacijskem procesu algoritma DEMO [201]. Hibridizacijo diferencialne evolucije in lokalnega preiskovanja smo izvedli tako, da SQP skuša izboljšati nekatere poskusne vektorje iz DE. Vektor iz SQP zamenja star populacijski vektor le, če ga dominira.

Sekvenčno kvadratično programiranje (SQP) [197, 16] je nelinearna optimizacijska metoda, ki sloni na izračunavanju gradienta in je ena od posplošitev Newtonove metode na več dimenzij. Z metodo SQP lahko rešujemo tudi tudi probleme z nelinearnimi omejitvami. Ko podamo začetno rešitev problema, metoda v vsaki nadaljnji iteraciji reši aproksimacijski model in se tako skuša približati rešitvi osnovnega problema. Osnovno kriterijsko funkcijo aproksimira z njenim kvadratičnim aproksimacijskim modelom in izračuna optimum aproksimiranega kvadratičnega modela. Optimum tega modela ni nujno tudi optimum osnovne kriterijske funkcije. Kot pri večini optimizacijskih metod, metoda SQP ni izvedena le kot en algoritem, temveč služi le kot osnutek, iz katerega so se razvili številni specifični algoritmi [16]. Algoritem za metodo SQP, ki smo ga uporabili, je FSQP-AL. Implementacijo najdemo v programskem paketu CfSQP [134][6].

[6]Uporabili smo programski paket CfSQP z akademsko licenco.

Hibridizacijo diferencialne evolucije v svojem algoritmu DEMOwSA in lokalnega preiskovanja SQP smo izvedli tako, da SQP skuša izboljšati nekatere poskusne vektorje iz DE. Najprej iščemo globalno rešitev z DE in dobimo poskusni vektor. Nato tega na vsakih $G_{\mathrm{LS}} = 10$ generacij z do $FEs_{\mathrm{LS}} = 5000$ ovrednotenji v lokalnem preiskovanju SQP skušamo izboljšati. Izboljšati skušamo do $ind_{\mathrm{LS}} = 0{,}15$ (to vrednost smo dobili empirično) naključno izbranih populacijskih vektorjev iz trenutne populacije. Izbiramo le posameznike, ki jih je predhodno izboljšal že globalni algoritem, saj se s tem izognemo nepotrebnemu večkratnemu lokalnemu preiskovanju istih vektorjev, kar bi po nepotrebnem povečalo število ovrednotenj kriterijske funkcije. Najprej izvedemo $miter_1 = 3$ iteracije SQP in če dobljen vektor dominira nov poskusni vektor iz DE, izvedemo še $miter_2 = 10$ iteracij nad izboljšanim vektorjem. Tak vektor, dobljen iz SQP, zamenja populacijski vektor v DE le, če ga dominira. Nov algoritem poimenujemo DEMOwSA-SQP. Pri SQP smo uporabili le omejitve za prostor spremenljivk, tj. linearni omejitvi minimuma in maksimuma za vsako spremenljivko. Konvergenčno stopnjo pri SQP smo nastavili na $\epsilon = 10^{-8}$. Stopnjo prilagajanja krmilnih parametrov v DE smo izbrali $\tau = 2/\sqrt{2D}$.

Algoritem DECMOSA-SQP – diferencialna evolucija za večkriterijsko optimizacijo z logaritmično normalnim samoprilagajanjem, lokalnim iskanjem in prilagajanjem dopustnosti omejitev

Algoritem DEMOwSA-SQP smo dopolnili še z obravnavo omejitev s prilagajanjem dopustnosti omejitev in novi algoritem poimenovali *Differential Evolution with Self-adaptation and Local Search for Constrained Multiobjective Optimization* (DECMOSA-SQP) in ga ovrednotili nad večkriterijskimi problemi z omejitvami iz svetovnega tekmovanja *Constrained Multiobjective Optimization* na CEC 2009 [265].

Hibridizacijo globalnega preiskovanja z lokalnim preiskovanjem nekaterih posameznikov v DECMOSA-SQP smo izvedli enako kot pri DEMOwSA-SQP, z manjšimi spremembami krmilnih parametrov hibridizacije obeh pristopov. Obseg izvajanja SQP smo namreč nekoliko zmanjšali, na vsakih $gen_{\mathrm{LS}} = 25$ generacij in za največ $ind_{\mathrm{LS}} = 0{,}1\%$ naključno izbranih posameznih populacijskih vektorjev.

Obravnavo omejitev smo v algoritmu DECMOSA-SQP vključili s spremembo izračuna dominantnosti v selekcijskem operatorju. Dopustnost omejitev smo prilagajali in izračunavali enako kot algoritem ϵ-jDE [20]. Slednji operator v algoritmu uporabimo na dveh mestih. Prvič, po vsaki iteraciji pri strategiji takojšnje posodobitve novega vektorja v trenutni populaciji in drugič, po vsaki generaciji pri rezanju populacijske množice nedominiranih vektorjev s strategijo SPEA2 [284].

Algoritem MOjDE – logistično samoprilagodljiva večkriterijska diferencialna evolucija

Algoritem $DEMO$ [201, 227] smo nadgradili tudi z mehanizmom samo-prilagajanja krmilnih parametrov iz algoritma jDE [23] in novo nastali algoritem poimenujemo $MOjDE$ [30]. Algoritem $MOjDE$ uporablja strategijo DE/rand/1/bin [215, 183, 74] z mehanizmom samoprilagajanja krmilnih parametrov iz algoritma jDE [23], ki smo ga nekoliko omejili za parameter CR:

$$CR_{i,g+1} = \begin{cases} CR_{\mathrm{l}} + rand_3 \times CR_{\mathrm{u}} & \text{če } rand_4 < \tau_2, \\ CR_{i,g} & \text{sicer.} \end{cases} \tag{2.43}$$

$rand_j, j \in \{3, 4\}$ sta naključni realni vrednosti na intervalu $[0, 1)$. τ_2 je verjetnost, s katero uravnavamo krmilna parametra F in CR. Konstante imajo naslednje vrednosti: $\tau_2 = 0{,}1$, $CR_{\mathrm{l}} = 0{,}1$ in $CR_{\mathrm{u}} = 0{,}6$. Nov CR ima naključno vrednost na intervalu $[0{,}1,\ 0{,}7)$. V začetni populaciji algoritma MOjDE imajo vsi posamezniki krmilni parameter $F_{\mathrm{init}} = 0{,}5$ in krmilni parameter $CR_{\mathrm{init}} = 0{,}7$. $DEMO$ ima podporo za več selekcijskih strategij ($NSGA\text{-}II$, $SPEA$, $IBEA$), pri algoritmu MOjDE smo uporabili selekcijsko strategijo $SPEA$.

Primerjavo algoritmov $DEMO$ in $MOjDE$ smo izvedli tudi na testnih funkcijah in s pomočjo orodij, ki so bila predlagana na posebni sekciji za oceno učinkovitosti večkriterijskih algoritmov za optimizacijo IEEE Congress on Evolutionary Computation (CEC 2007) [98] ter rezultate objavili v prispevkih [30, 261].

Algoritem $\mathrm{jDE_{NP,MM}}$ – algoritem DE z zmanjševanjem velikosti populacije in več mutacijskih strategij

Nadgradili smo tudi eno od razširitev algoritma jDE [23], algoritem dynNP-DE [25]. Ta obravnava zmanjševanje populacije med iskanjem, za katero sta Neri in Tirronen pokazala [166], da se izkaže za še posebej kakovostno razširitev algoritma jDE. V dynNP-DE smo tako ob zmanjševanju populacije vključili še dve mutacijski strategiji, rand/1/bin in best/1/bin. Prvo strategijo smo izvajali, ko je bila populacija večja od 100, sicer v polovici primerov. Drugo strategijo smo izvajali v preostalih primerih. Velikost začetne populacije (NP) smo nastavili na 200 in število delitev populacije ($pmax$) na 4. Opisan algoritem imenujemo $\mathrm{jDE_{NP,MM}}$ in smo ga skupaj z meritvami kakovosti na realnih industrijskih izzivih predstavili v prispevku [258].

2.2 Proceduralno modeliranje dreves

Modeliranje narave v abstraktne umetne sisteme je ena od temeljnih dejavnosti na vseh področjih znanosti o naravi[7], saj nam omogoča njene gradnike predstaviti na formalen, omejen in od neupoštevanega okolja ločen način. Razumevanje naravnih zakonitosti nam pomaga aproksimativno posnemati model, ki do določene mere sledi naravi. Z modeliranjem narave in okolja okoli sebe se spopade vsak senzorski sistem, ki ima krmilno zanko [135], v kar lahko štejemo tudi vsako živo bitje z možnostjo odzivanja na okolje. Celični organizmi, večcelični organizmi, rastline, živali in podobna živa bitja, tudi Homo Sapiens Sapiens, v svojih celicah gradijo model svojega naravnega okolja. Najvidnejši organ je morda živčni sistem oz. možgani, ki vsebuje veliko število nevronov, ki krmilijo tudi mišljenjski proces. V procesu mišljenja je človek skozi izkušnje pripravil več modelov naravnih sistemov, en od takšnih je tudi struktura naravnih bitij, kot so olesenele rastline (drevesa). Njihovo strukturo je človek prvo modeliral pri spoznavanju okolice, gibanju in bivanju, nato pri slikanju. Na slikah več civilizacij opazimo upodobitve dreves. Po obdobju t.i. srednjega veka evropske zgodovine se je z modeliranjem drevesne strukture ukvarjal Da Vinci; poleg razsvetljenskih in antičnih znanosti je bil tudi učenec zgodnjih azijskih ved in hermeticizma, v katerih je v arhitekturni in likovni znanosti že razširjena uporaba zlatega reza naravne samopodobnosti [213]. Da Vinci je postavil cevni model drevesne strukture [211], ki vsebuje postopek za opis nekaterih opaženih zakonitosti v rekurzivno samopodobni drevesni strukturi in je tako prvi proceduralni model opisa drevesne strukture. Relacijo med debelinami izhajajočih vej je z ideogrami kalkulusa [94, 168] natančneje formaliziral Mandelbrot [148], ki je definiral Pitagorov izrek med debelino osnovne veje in debelinami obeh stranskih vej.

Danes obstaja več modelov za opis trodimenzionalnega izgleda dreves. En od teh je ročna postavitev drevesne strukture in listov s splošnonamenskim geometrijskim modelirnikom, kar je zamudno, saj je potrebno vse gradnike drevesa ročno določiti z osnovnimi geometrijskimi ploskvami. Zato raje uporabimo proceduralne modele za postopkovni izračun drevesne geometrije. Poznamo več proceduralnih modelov, zato podajmo njihove glavne predstavnike in njihove ključne uvedene razlikovalne značilnosti. Temeljna značilnost proceduralnih modelov je gradnja osnovne vejitvene strukture [187]. Nekateri modeli omogočajo prilagajanje stopnje podrobnosti [5, 15, 195], drugi so bolj fleksibilni in prirejeni za učinkovitejšo uporabo [96, 237] ali prostorsko

[7]Narava, s tujko fizika, starogrško $\varphi\acute{\upsilon}\sigma\iota\varsigma$, *physis*; prva omemba v Odiseji [175].

učinkoviti [173, 216]. Modeli nudijo različne tipe animacije, saj končno zgrajen model predstavijo različno, od stožcev in prizem [96, 237], do fraktalov [173, 216]. Večina teh modelov skuša določiti lastnosti izgleda končnega trodimenzionalnega modela drevesa, ki je pogosto sestavljen iz ploskev ali kar trikotnikov v prostoru. Z geometrijsko projekcijo iz trodimenzionalnega prostora drevesa nato najpogosteje upodobimo na dvodimenzionalno ploskev, pri čemer odstranimo eno dimenzijsko komponento trodimenzionalnega modela. Proceduralni modeli določajo morfološke lastnosti, kot je rotacija vej okoli centralne osi rasti. Te lastnosti so pogosto biološko osnovane na filotaksi, t.j. eni od glavnih genetskih značilnosti celic v drevesih, ki določa smer rasti drevesne arhitekture [196],

Enega od prvih specializiranih modelov za tvorbo dreves na računalniku za namen trodimenzionalne animacije sta predstavila Aono in Kunii [5]. Razvila sta štiri geometrijske modele, ki predstavljajo zaporedne nadgradnje. Skupna so jim pravila, da vsaka vejitev osnovne veje tvori dve podveji, da se dolžina in premer podvej manjšata s konstantnim faktorjem, da so koti vejitev enaki na vseh nivojih drevesa, da je ravnina, ki jo določata podveji, pravokotna na ravnino, ki jo podajata starševska in prastarševska veja, ter da se vejitve hkrati izvedejo na vrhu vseh obstoječih vej. Opisala sta tudi kot osnega zasuka zaporednih stranskih vej vzdolž starševske veje, to lastnost imenujemo filotaksa [196]. V modele sta vključila točkaste atraktorje, ki pritegnejo ali odbijejo rast drevesa in simulirajo učinek vetra, sonca in gravitacije.

Bloomenthalov model [15] na osnovi ročno podanega skeleta drevesa zgolj doda sedla pri vejitvah vej in skelet obleče s krivuljami NURBS, ki jih opremi s foto-teksturami.

Reeves in Blau sta za predstavitev gozdnih površin pri manjši podrobnosti uporabila sisteme delcev [195]. Veje dreves sta predstavila z daljicami, liste s točkami (majhnimi krogi), osnovna debla dreves s stožci. Parametri njunega modela drevesa so širina spodnjega dela krošnje drevesa, višina drevesa, višina od tal do prve veje na krošnji, srednja dolžina vej in kot vejitve. Iz algoritma dobljena vejitvena struktura se naknadno obdela z ločenimi algoritmi za simuliranje gravitacije, vpliv vetrov in težnje po svetlobi.

Oppenheimer je drevesa modeliral z uporabo fraktalnih tehnik [173]. Deblo in veje gradi z izvajanjem linearnih transformacij, podanih z matrikami velikosti 3×3. Geometrijo drevesa sestavljajo mnogokotniške prizme ali le daljice vzdolž vej, na katere je nalepljena proceduralna tekstura. Upodobljeni modeli so namenjeni visoki stopnji podrobnosti, vendar ne vsebujejo listov drevesa.

Z biološko motiviranim modelom z L-sistemi [140] je Prusinkiewicz modeliral drevesa pri visoki stopnji podrobnosti [187]. V L-sisteme je vpeljal grafično predstavitev prepisnih nizov, definiranih s kontekstno odvisno gramatiko. Prepisne nize je vizualiziral po principu želve Logo [174], ki je z geometrijsko interpretacijo niza narisala topologijo drevesa. Ta tehnika ima veliko izrazno moč in je doživela največ dopolnitev [185, 154, 186, 62, 142, 61, 131, 203, 40, 234]. Realizacija modelirnika je zato precej zahtevna in modeliranje drevesa zahteva dobro poznavanje uporabljenega domensko specifičnega jezika za definicijo gramatike prepisnega niza. V ta namen je večji poudarek na interaktivnosti oblikovanja in upodabljanja [61, 142] kot na novih predstavitvenih metodah.

Holtonov žilni model [96] je prav tako biološko motiviran, saj debeline vej in razmerja vejitev od osnovne veje določa žilni model. Model temelji na omenjenem Da Vincijevem cevnem modelu [211]. Porazdelitev žil v drevesu predstavlja tok cevi Da Vincijevega modela in določa debelino ter posledično dolžino veje, saj se žile celoštevilsko delijo v podveje, veje z eno žilo pa imajo liste. Modelu poleg števila vseh žil v drevesu podamo še razmerja med dolžinami vej in kote vejitev med izhajajočimi vejami. Na kote vejitve vplivajo tudi elementi okolja, ki določajo težnje drevesa, kot so pokončnost debla oz. uravnoteženost krošnje, upogibanje vej k tlom zaradi vpliva gravitacije, rast v smeri svetlobe, rast v navpični smeri, rast v vodoravni smeri in rast v ravnini, ki je pravokotna na ravnino med osrednjo osjo drevesa in starševsko vejo. Dobra stran tega modela je omenjeni samodejni izračun debeline in razmerja med delitvami vej, vendar mora uporabnik preostale parametre še vedno vnesti tabelarično, kar zmanjša okretnost modeliranja.

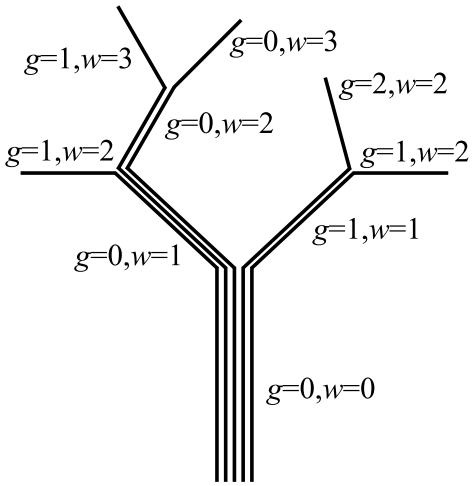
Weber in Penn sta drevo predstavila s preprosto geometrijo [237] brez razvoja vejitvene strukture. Za vse veje na enakih nivojih sta podala kot vejitve, razmerje dolžin glede na osnovno vejo in debeline vej. Predstavila sta tudi možnost animacije zibanja drevesa v vetru, obrezovanje vej drevesa na določen volumen in poenostavljanje geometrije drevesa.

Strnad je drevesa za upodobitev pri srednji podrobnosti predstavil s fraktalnimi hiperteksturami [216]. Drevesa je definiral fraktalno z iterativnimi funkcijskimi sistemi, ki jih je upodobil s prostorskim upodabljanjem. Model zaradi časovno zamudne tehnike upodabljanja ni primeren za uporabo v interaktivni animaciji, vendar omogoča zvezen prehod med stopnjami podrobnosti.

Pri vseh naštetih tehnikah proceduralnega modeliranja je potrebno doseči kompromis med okretnostjo in kompleksnostjo modeliranja. Nekateri splošno namenski geometrijski modelirniki danes že vključujejo specialna orodja za izdelavo dreves. Problem teh orodij je v tem, da so premalo okretna in uporabniku omogočajo le omejen nabor tipičnih predstavnikov drevesnih vrst ali pa zaradi želje po čim večji oblikovalski svobodi definirajo veliko število numeričnih parametrov, katerih nastavljanje je zamudno in ne-intuitivno, pomen pa pogosto nejasen. Zato smo si postavili cilj izdelati geometrijski model drevesa z minimalnim številom potrebnih parametrov, katerih nastavljanje bo intuitivno in interaktivno. Model naj bi bil dinamičen in tako omogočal enostavno animacijo. To nam je uspelo s kombinacijo obstoječih tehnik, premišljenim izborom disjunktivnih numeričnih značilnosti dreves in grafičnim vmesnikom za podajanje vrednosti parametrov, katerih spreminjanje je interaktivno vidno na senčenem modelu drevesa. Rezultat je modelirnik dreves, ki vključuje tudi sistem za simulacijo rasti in animacijo dreves. Naš modelirnik temelji predvsem na Holtonovem žilnem modelu dreves in uporablja nekaj idej iz Weber-Pennovega modela. Zato bomo najprej podrobneje predstavili oba modela.

Za modeliranje drevesa v tej disertaciji smo uporabili numerično kodirani proceduralni model iz programskega sistema EcoMod, prvič predstavljenega v [271] in natančneje opisanega v [268]. Parametriziran proceduralni model zgradi prostorsko trodimenzionalno strukturo drevesa z izvajanjem fiksnega postopka oz. procedure nad danim naborom numerično kodiranih vhodnih parametrov, kot so npr. širina debla, relativna dolžina vej in vejitvena struktura. Ta procedura rekurzivno [152] izračunava sestavne dele drevesa. Drevo lahko modeliramo interaktivno ali pa z nalaganjem parametrov iz knjižnice že oblikovanih dreves, ki je priložena modelirniku. Pri tem se vizualizacija osveži takoj, ko spremenimo kak parameter modela, kar pripomore pri učenju uporabe modelirnika. Parametriziran proceduralni model lahko kasneje uporabimo za računalniško animacijo, saj s spremembo glavnih ali pomožnih parametrov modela dobimo podobne, le nekoliko spremenjene modele, ki so primerni za izračunavanje ključnih animacijskih okvirjev [269].

Izdelan model se od omenjenih modelov [5, 15, 96, 237, 154, 195, 173, 203, 148, 216, 187, 185, 186, 142, 62, 61, 167, 41, 192, 204, 117, 40] razlikuje v tem, da je popolnoma numerično kodiran in je nespremenljive dimenzije. To nam omogoča uporabo diferencialne evolucije, ki se dobro obnese nad problemi z velikimi dimenzijami [220, 52], kar smo preskusili tudi sami [263].



Slika 2.4: Graveliusov in Weibullov red veje.

2.2.1 Holtonov žilni model dreves

Holtonov model [96] temelji na notranji žilni strukturi botaničnih dreves. Deblo definira kot prvi del drevesa, ki prihaja iz zemlje, do prve vejitve ne glede na debelino vejitve. Glede na podano začetno število žil S v deblu drevesa se te pri vsaki vejitvi delijo na dve podveji, tako da se skupno število žil ohrani. Število odsekov vej B, ki jih dobimo zaradi vejitev, je:

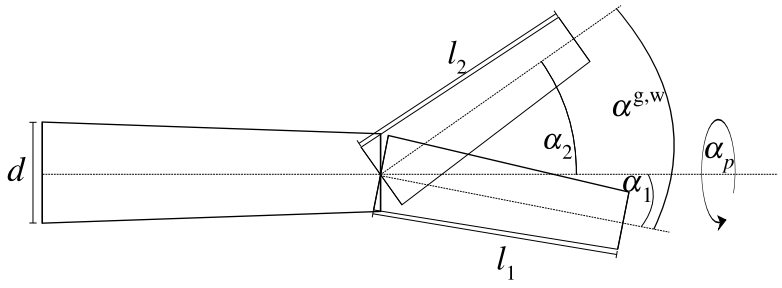
$$B = 2S - 1, \tag{2.44}$$

kjer število žil S posredno določa ostale lastnosti veje, kot sta npr. njena debelina in dolžina. S številom žil je omejeno tudi število nadaljnjih vejitev, ki se končajo pri veji z eno žilo, iz katere rastejo le še listi. Na sliki 2.4 je $S = 5$ in $B = 9$.

Drug pomemben podatek, ki ga beležimo pri vsaki veji, je njen *red*. Vendar tudi pri tem ločimo dve različni obliki štetja, ki določata Graveliusov oz. Weibullov red. Graveliusov red veje označimo z g, Weibullov red pa z w. Pri Graveliusovem štetju se ob vsaki cepitvi veje t.i. *stranski podveji* red poveča, medtem ko druga, t.i. *glavna podveja*, šteje kot nadaljevanje prvotne ali *osnovne veje* in s tem ohrani njen red. Pri Weibullovem štetju se obema podvejama red poveča (slika 2.4).

Holton za izračun geometrije drevesa uporablja biološko osnovane enačbe, s pomočjo katerih določa ostale lastnosti veje s pomočjo (v odvisnosti od) števila žil v njej. V enačbah pogosto nastopajo uporabniško nastavljivi parametri, ki so definirani ločeno za vsak red veje (posebej Graveliusovega in Weibullovega). Vsi parametri Holtonovega modela so:

- S – število žil v deblu drevesa, ki določa njegovo kompleksnost in s tem tudi starost,
- $k_{\mathrm{s}}^{g,w}$ – razmerje porazdelitve žil na podveji pri vejitvah,
- $\alpha^{g,w}$ – kot med izhajajočima podvejama pri delitvi,
- $l_0^{0,0}$ – višina osnovnega debla,
- k_{d} – koeficient debeline veje,
- $m_t^{g,w}$ in $M_t^{g,w}$ – spodnja in zgornja meja relativne dolžine podvej glede na dolžino osnovne veje, ločeno za tip podveje ($t \in \{major, minor\}$),
- $N_t^{g,w}$ in $n_t^{g,w}$ – spodnja in zgornja meja dejanske dolžine podvej glede na dolžino osnovne veje, ločeno za tip podveje ($t \in \{major, minor\}$),
- k_{c}^{g} – gravicentralizem (t.j. težnja debla po navpični rasti), skalirni faktor med 0 in 1 za vektor v smeri popravka rasti proti centru (če je vrednost prevelika, drevo začne nihati v levo in spet desno okoli središčne osi debla),
- $\alpha_{\mathrm{m}}^{g,w}$ – gravimorfizem (t.j. upogibanje vej zaradi gravitacije),
- k_{p} – fototropizem (tendenca rasti v smeri proti svetlobi),
- $k_{\mathrm{o}}^{g,w}$ – ortotropizem (t.j. tendenca po rasti vej navpično navzgor),
- $k_{\mathrm{p}}^{g,w}$ – plagiotropizem (t.j. tendenca po rasti vej horizontalno navzven od debla),
- α_{p} – kot filotakse (t.j. tendenca po rasti veje izven ravnine sestrske veje); s tem je določen tudi planartropizem,
- k_{w} – koeficient popačenja za naključnost rasti in
- $\overline{\mathbf{k}}$, $\overline{\mathbf{p}}$, $\overline{\mathbf{h}}$, $\overline{\mathbf{g}}$, $\overline{\mathbf{w}}$, $\mathbf{b_0}$, $\mathbf{q}$, $\mathbf{b}$ – enotski vektorji v smeri ortotropizma, plagiotropizma, fototropizma, gravimorfizma, vektor popačenja, začetni vektor veje, smer veje ter dejanski vektor veje.



Slika 2.5: Izračun geometrije pri vejitvi.

Smer glavne podveje tako izračunamo s prištevanjem vektorjev tendenc k smeri osnovne veje:

$$\mathbf{q} = \mathbf{b_0} + k_{\mathrm{s}}^{g,w}\overline{\mathbf{k}} + k_{\mathrm{p}}^{g,w}\overline{\mathbf{p}} + \alpha^{g,w}\overline{\mathbf{h}} + \alpha_{\mathrm{m}}^{g,w}\overline{\mathbf{g}} + k_{\mathrm{w}}\overline{\mathbf{w}}. \tag{2.45}$$

Smer glavne podveje in njeno dolžino določimo z naslednjim vektorjem:

$$\mathbf{b} = \frac{l_1\mathbf{q}}{\|\mathbf{q}\|}, \tag{2.46}$$

kjer je l_1 dolžina glavne podveje (glej enačbo 2.56).

Debelino veje d izračunamo iz števila žil S_0:

$$d = k_{\mathrm{d}}\sqrt{S_0}, \tag{2.47}$$

kjer je parameter $k_{\mathrm{d}} \in [0, 1]$ koeficient debeline, ki je konstanten za celotno drevo.

Ob vsaki vejitvi se žile osnovne veje razdelijo med nastali podveji:

$$S_1 = \lceil 1 + k_{\mathrm{s}}^{g,w}\,(S_0 - 2)\rceil\,, \tag{2.48}$$

$$S_2 = S_0 - S_1. \tag{2.49}$$

Parameter $k_{\mathrm{s}}^{g,w} \in \left[\frac{1}{2}, 1\right]$ določa delež žil, ki gredo v glavno podvejo. Pri tem je S_0 število žil osnovne veje, S_1 število žil v glavni podveji in S_2 število žil v stranski podveji.

Po vejitvi se podveji odklonita vsaka na svojo stran v isti ravnini, tako da oklepata uporabniško definirani kot vejitve $\alpha^{g,w} \in [0°, 180°]$ (slika 2.5). Zatem se obe podveji zasučeta okoli vzdolžne osi osnovne veje za kot $\alpha_{\mathrm{p}} \in [0°, 180°]$. Pri tem se upošteva še koeficient gravicentralizma $k_{\mathrm{c}}^{g,w} \in [0, 1]$, ki je različen od 1 samo za veje Graveliusovega reda 0. Odklonska kota α_1 za glavno podvejo in α_2 za stransko podvejo tako izračunamo kot:

$$\alpha_1 = k_{\mathrm{c}}^{g,w}\sqrt{\frac{S_2}{S_0}}\alpha^{g,w}, \tag{2.50}$$

$$\alpha_2 = \alpha^{g,w} - \alpha_1. \tag{2.51}$$

L_0, L_1 in L_2	relativne dolžine starševske veje in obeh podvej
R_1 in R_2	razmerji relativnih dolžin podvej glede na starševsko vejo
l_1 in l_2	dejanski dolžini podvej
r_1 in r_2	razmerji dejanskih dolžin podvej glede na starševsko vejo
$m^{g,w}_{\text{minor}}$ in $M^{g,w}_{\text{minor}}$	spodnja in zgornja meja relativne dolžine stranske podveje
$m^{g,w}_{\text{major}}$ in $M^{g,w}_{\text{major}}$	spodnja in zgornja meja relativne dolžine glavne podveje
$n^{g,w}_{\text{minor}}$ in $N^{g,w}_{\text{minor}}$	spodnja in zgornja meja dejanske dolžine stranske podveje
$n^{g,w}_{\text{major}}$ in $N^{g,w}_{\text{major}}$	spodnja in zgornja meja dejanske dolžine glavne podveje

Tabela 2.2: Strukturni parametri žilnega modela.

Gravimorfizem je vključen z dodatnim zasukom veje v smeri tal, pri čemer je kot tega zasuka podan s parametrom $\alpha^{g,w}_{\text{m}} \in [0°, 180°]$.

Za dolžine vej (pomen uporabljenih oznak je zbran v tabeli 2.2) Holton najprej izračuna razmerji žil med osnovno vejo in podvejama:

$$R_1 = r_1 = \sqrt{\frac{S_1}{S_0}}, \qquad R_2 = r_2 = \sqrt{\frac{S_2}{S_0}}. \tag{2.52}$$

Razmerji je nato omejil:

$$m^{g,w}_{\text{major}} \leq R_1 \leq M^{g,w}_{\text{major}}, \qquad m^{g,w}_{\text{minor}} \leq R_2 \leq M^{g,w}_{\text{minor}} \tag{2.53}$$

in izračunal specifične dolžine vej:

$$L_1 = R_1 L_0, \qquad L_2 = R_2 L_0, \tag{2.54}$$

ki jih je še skaliral glede na uporabniško nastavljiva razmerja:

$$n^{g,w}_{\text{major}} \leq r_1 \leq N^{g,w}_{\text{major}}, \qquad n^{g,w}_{\text{minor}} \leq r_2 \leq N^{g,w}_{\text{minor}} \tag{2.55}$$

in tako dobil dejanske dolžine vej:

$$l_1 = r_1 L_0, \qquad l_2 = r_2 L_0. \tag{2.56}$$

Holton poda intervale uporabnih vrednosti za posamezne parametre drevesa in ugotavlja, da začetno število žil med 2000 in 8000 daje najboljše rezultate. Kot primer podajmo, da je jelke (slika 2.6) modeliral z naslednjimi parametri: $S = 2500$, $k^{0,w}_{\text{s}} = 0{,}985$,

Slika 2.6: Holtonova upodobitev parametričnega modela jelke [96].

$k_{\mathrm{s}}^{i,w} = 0{,}6$, $\alpha^{0,w} = 115°$, $\alpha^{i,w} = 35°$, $M_{\mathrm{major}}^{0,w} = 1$, $m_{\mathrm{major}}^{0,w} = 0{,}98$, $M_{\mathrm{major}}^{i,w} = 0{,}98$, $m_{\mathrm{major}}^{i,w} = 0{,}85$, $M_{\mathrm{minor}}^{0,w} = 0{,}76$, $m_{\mathrm{minor}}^{0,w} = 0{,}7$, $N_{\mathrm{major}}^{0,w} = 0{,}15$, $n_{\mathrm{major}}^{0,w} = 0{,}1$, $N_{\mathrm{major}}^{i,w} = 0{,}9$, $n_{\mathrm{major}}^{i,w} = 0{,}8$, $i = 1, 2, 3, ...$ in $w = 0, 1, 2, ...$ Ostali parametri niso podani (α_{p}, $M_{\mathrm{minor}}^{i,w}$, $m_{\mathrm{minor}}^{i,w}$, $N_{\mathrm{minor}}^{g,w}$, $n_{\mathrm{minor}}^{g,w}$) in zato pri gradnji geometrijskega modela pripadajoči operatorji niso uporabljeni. Za vse tri drevesa je uporabil enake parametre, vsaka pa ima drugo seme za generator psevdo-naključnih števil za vektorje popačenja. Dobljene skelete dreves je oblekel in upodabljal s krivuljami NURBS.

2.2.2 Weber-Pennov model

Podoben pristop kot Holton sta uporabila Weber in Penn, ki sta definirala 40 različnih numeričnih parametrov drevesa [237]. Pri tem so nekateri (npr. kot vejitve) definirani ločeno za veje reda 1, 2 in 3 ali več, tako da je njihovo dejansko število preko 80. Definirala sta tudi poenostavljanje geometrije, ki glede na razdaljo od gledišča izriše manj redov vej. Veje sta upodabljala s stožci. Definirala sta tudi nekaj več funkcionalnosti kot Holton, kot je zibanje drevesa v vetru, stopnjo podrobnosti in obrezovanje drevesa na podan obris. Pozibavanje v vetru sta definirala kot nagib veje za naključen vektor v smeri vetra, ki ga s časom spreminjamo. Vsaka veja v njunem modelu ima svoje seme za generator psevdo-naključnih števil za določitev začetnega kota zibanja.

V našem modelirniku EcoMod smo iz Weber-Pennovega modela vzeli idejo o predstavitvi vej s stožci, animacijo drevesa pod vplivom vetra in možnost poenostavljanja geometrije na nivoju proceduralnega modela. Weber in Penn sta red veje omejila na 5, kar nam je dalo idejo, da bi parametrizacijo Holtonovega modela opravili prav tako z omejenim številom nivojev, ki bi jih bilo možno interaktivno načrtovati z grafi.

2.2.3 Zasnova interaktivnega modelirnika dreves EcoMod

Za upodabljanje dreves pri eksperimentih za to disertacijo uporabimo naš geometrijski proceduralni model EcoMod [273]. Posamezno drevo modeliramo s postavljanjem parametrov znotraj proceduralnega modela. Parametre v drevesnem modelu, podobno kot Holton [96], ločimo na vektorske in skalarne, pri čemer lahko vektorske interaktivno oblikujemo z grafi. Med vektorske parametre proceduralnega modela štejemo strukturo vejitev, kompleksnost zgradbe, kote vejitev, razmerja med dolžinami odsekov vej in vpliv gravitacije na geometrijo drevesa. Skalarni parametri v modelu so višina in debelina osnovnega debla, vpliv vetra ter gostota in velikost listov. S pomočjo navedenih vektorskih in skalarnih parametrov rekurzivno zgradimo geometrijski model, ki mu dodamo teksture za končni izgled drevesa. Pri določanju nabora parametrov dreves, ki smo jih v modelirniku EcoMod podprli, smo upoštevali naslednja načela:

- pomen vseh parametrov mora biti jasen in njihov učinek čim bolj predvidljiv,
- nabor parametrov naj bo dovolj raznolik, da omogoča modeliranje večine v naravi najpogosteje zastopanih drevesnih oblik,
- parametri naj nimajo podvojenega ali konfliktnega delovanja in
- nabor parametrov naj vključuje lastnosti drevesa, ki so pomembne pri animaciji.

Iz Holtonovega modela smo tako odstranili manj pomembne parametre in dodali nekatere ideje iz Weber-Pennovega modela za podporo stopnji podrobnosti in animaciji pozibavanja v vetru. V modelirniku so vgrajeni vsi parametri našteti v razdelku 2.2.1 razen ortotropizma $k_{\mathrm{o}}^{g,w}$ (modeliran z negativnim gravimorfizmom), fototropizma k_{p}, plagiotropizma $k_{\mathrm{p}}^{g,w}$ in koeficienta popačenja k_{w} (pri upodabljanju ekosistema [269] nadomeščen s sukanjem drevesa okoli osi debla), saj želimo parametre čim bolj poenostaviti. Tudi relativne dolžine vej $M_t^{g,w}$ in $m_t^{g,w}$ modeliramo skupno za oba tipa vej, dejanske dolžine $N_t^{g,w}$ in $n_t^{g,w}$ pa zamenjamo s skalirnim faktorjem dolžine veje $k_{\mathrm{l}}^{g,w}$. Način porazdelitve listov l_{type} in gostota listov ρ_{l} sta povzeti iz orodja Corel Bryce [232]. Po uvedenih spremembah so v naš model dodani naslednji parametri:

- $M^{g,w}$ – zgornja meja relativne dolžine podvej glede na dolžino osnovne veje (enaka za oba tipa podvej, odstranimo parameter $t \in \{minor, major\}$),

- $m^{g,w}$ – spodnja meja relativne dolžine podvej glede na dolžino osnovne veje (enaka za oba tipa podvej, odstranimo parameter $t \in \{minor, major\}$),
- $k_{\mathrm{l}}^{g,w}$ – skalirni faktor dolžine veje,
- l_{type} – način porazdelitve listov,
- ρ_{l} – gostota listov,
- l_{l} – velikost listov,
- t_{bark} (t_{leaf}) – identifikatorja teksture debla (listov),
- l_{LOD} – stopnja poenostavitve geometrijske strukture drevesa,
- w_{s} – jakost neusmerjenega vetra na drevo,
- k_{f} – prožnost vej drevesa pri vetru,
- w_{g} – jakost usmerjenega vetra na drevo in
- $\mathbf{w}$ – smer usmerjenega vetra na drevo.

Glede na to, ali parameter opisuje lastnost drevesa, ki je enaka za celotno drevo, ali pa lastnost, ki se spreminja z lego v drevesu, podobno kot Holtonov model, ločimo dve vrsti parametrov. Prvi parametri so globalni in od reda veje (g, w) neodvisni, drugi parametri so lokalni in od reda veje (g, w) odvisni. Pri tem so skalarni parametri globalni, konstantni za vse odseke vej v drevesu. Red veje označimo z Graveliusovim (g) in Weibullovim (w) indeksom veje, kot indeksni par (g, w). Poudariti moramo, da globalnost tu velja za eno drevo.

Globalne lastnosti (parametre) drevesa (npr. velikost listov) in začetne vrednosti za tvorbo modela (npr. začetno število žil) interaktivno določimo s pomočjo pogovornega okna (slika 2.7), katerih vrednosti lahko vnesemo z vpisom števila ali s pomočjo grafičnih drsnikov. Ob spreminjanju parametrov se takoj osveži tudi prikaz drevesa. Drsniki omogočajo vnos vrednosti naslednjih parametrov:

- število žil S med 0 in 5000,
- višino prvega odseka debla $l_0^{0,0}$ med 0 m in 10 m,
- koeficient debeline veje k_{d} med 0 in 0,05,

Slika 2.7: Pogovorno okno za interaktivno nastavljanje globalnih parametrov parametričnega modela drevesa.

- gravicentralizem k_{c} med 0 in 1,
- kot filotakse α_{p} med 0° in 360°,
- gostoto listov ρ_{l} med 0 in 30,
- velikost listov l_{l} med 0 m in 0,3 m ter
- hitrost vetra w_{g} med 0 in 10.

V istem pogovornem oknu iz padajočega menija izberemo tudi tip porazdelitve listov l_{type} z vrednostmi `Spiral`, `Stacked`, `Staggered`, `Bunched` in `Coniferous`.

Od reda veje (g, w) odvisni parametri so matrični parametri in so lokalni glede na posamezen odsek veje. Matrične parametre tako naslavljamo z redom veje, kjer sta $g \in \{0, ..., 15\}$ in $w \in \{0, ..., 50\}$ celi števili, ki naslavljata 750 realno kodiranih skalarnih vrednosti matričnega parametra. Takšni matrični parametri so: porazdelitev žil $k_{\mathrm{s}}^{g,w}$, kot vejitve $\alpha^{g,w}$, omejitve dejanske dolžine vej $M^{g,w}$ in $m^{g,w}$, skalirni faktor vej $k_{\mathrm{l}}^{g,w}$ ter vpliv gravimorfizma $\alpha_{\mathrm{m}}^{g,w}$. Za opis od reda veje (g, w) odvisnega parametra uporabimo dva pomožna parametra, ki ju podamo grafično. Prvi parameter je odvisen od Graveliusovega reda ($g \in \{0, ..., 15\}$), drugi pa od Weibullovega reda ($w \in \{0, ..., 50\}$). Mejni vrednosti $g = 15$ in $w = 50$ sta izkustveno določeni in zadostujeta za načrtovanje dreves z deset in več tisoč žilami. Od reda (g, w) odvisne parametre iz krmilne

lomljenke (ang. *control polygon* [87]) vzorčimo samo na celoštevilskih vrednostih g in w, čeprav sta oba pomožna parametra podana z zvezno krmilno lomljenko. Da dobimo vrednost nastavljanega parametra pri indeksu (g, w), vrednost iz grafa za Graveliusovo porazdelitev pri indeksu g zmnožimo z vrednostjo iz grafa za Weibullovo porazdelitev pri indeksu w. Zmnožek pomnožimo še s koeficientom, ki je vezan na enoto in definicijsko območje izbranega matričnega parametra: za porazdelitev žil $k_{\mathrm{s}}^{g,w}$ je to delež žil (med 0 in 1), kot vejitve $\alpha^{g,w}$, število stopinj (med 0 in 180), dolžini odsekov vej $M^{g,w}$ in $m^{g,w}$, metri (med 0 m in 4 m), skaliranje dolžine odsekov vej $k_{\mathrm{l}}^{g,w}$, skalirni faktor (med 0 in 4) in vpliv gravimorfizma $\alpha_{\mathrm{m}}^{g,w}$, število stopinj (med -180° in 180°). Vrednost za nekatere matrične parametre še omejimo med najnižjo in najvišjo definicijsko vrednost. Od reda veje (g, w) odvisne lastnosti drevesa iz grafov v našem interaktivnem modelirniku tako dobimo s konvolucijami vektorjev:

$$k_{\mathrm{s}}^{g,w} = \max\left\{\min\left\{k_{\mathrm{s}}^{g} k_{\mathrm{s}}^{w}, 1\right\}, \frac{1}{2}\right\}, \tag{2.57}$$

pri čemer je $k_{\mathrm{s}}^{g,w} \in [\frac{1}{2}, 1]$ definiran s pomožnima parametroma $k_{\mathrm{s}}^{g} \in [\frac{1}{2}, 1]$ in $k_{\mathrm{s}}^{w} \in [0, 2]$,

$$\alpha^{g,w} = \min\left\{\alpha^{g}\alpha^{w}, 180°\right\}, \tag{2.58}$$

pri čemer je $\alpha^{g,w} \in [0°, 180°]$ definiran z $\alpha^{g} \in [0°, 180°]$ in $\alpha^{w} \in [0, 2]$,

$$\alpha_{\mathrm{m}}^{g,w} = \max\left\{\min\left\{\alpha_{\mathrm{m}}^{g}\alpha_{\mathrm{m}}^{w}, 180°\right\}, -180°\right\}, \tag{2.59}$$

pri čemer je $\alpha_{\mathrm{m}}^{g,w} \in [-180°, 180°]$ definiran z $\alpha_{\mathrm{m}}^{g} \in [-180°, 180°]$ in $\alpha_{\mathrm{m}}^{w} \in [0, 2]$,

$$M^{g,w} = M^{g} M^{w}, \tag{2.60}$$

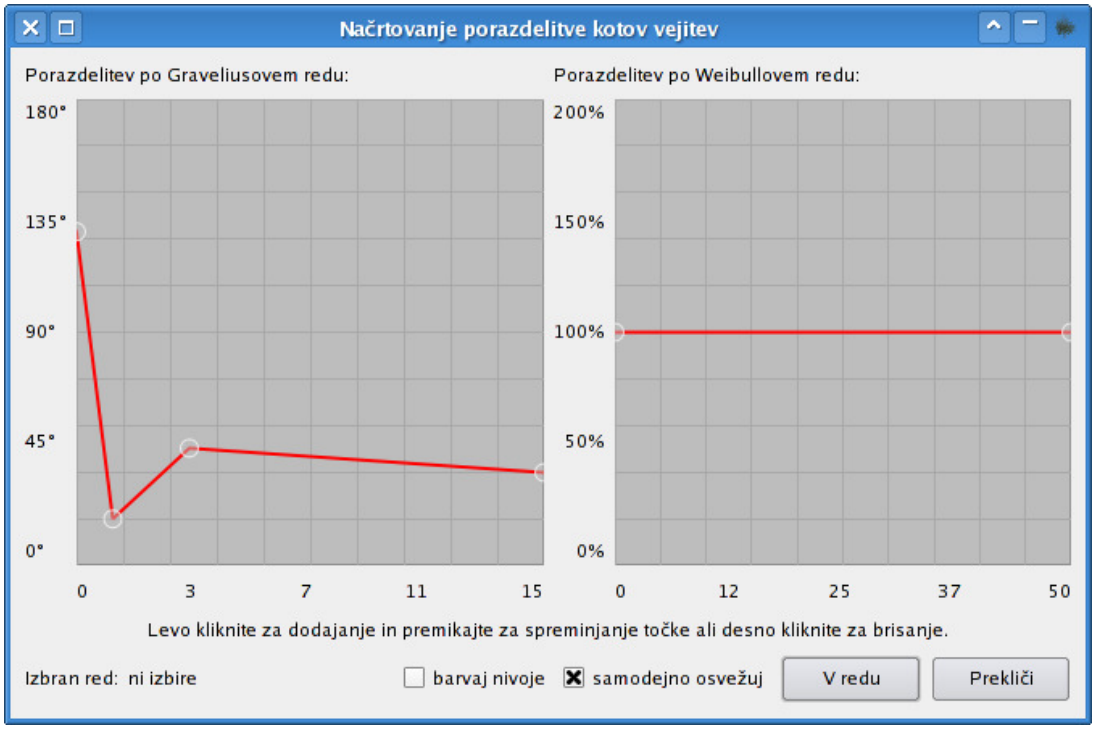
pri čemer je $M^{g,w} \in [0, 20]$ definiran z $M^{g} \in [0, 10]$ in $M^{w} \in [0, 2]$,

$$m^{g,w} = m^{g} m^{w}, \tag{2.61}$$

pri čemer je $m^{g,w} \in [0, 20]$ definiran z $m^{g} \in [0, 10]$ in $m^{w} \in [0, 2]$,

$$k_{\mathrm{l}}^{g,w} = k_{\mathrm{l}}^{g} k_{\mathrm{l}}^{w}, \tag{2.62}$$

pri čemer je $k_{\mathrm{l}}^{g,w} \in [0, 20]$ definiran s $k_{\mathrm{l}}^{g} \in [0, 10]$ in $k_{\mathrm{l}}^{w} \in [0, 2]$. Vrednosti navedenih lokalnih pomožnih parametrov uporabik poda z modeliranjem krmilne lomljenke, ki poteka grafično s pogovornim oknom (slika 2.8). Lomljenki lahko na poljubnem mestu dodamo novo krmilno točko, s čemer pripadajoči linearni odsek razpade na dva nova.

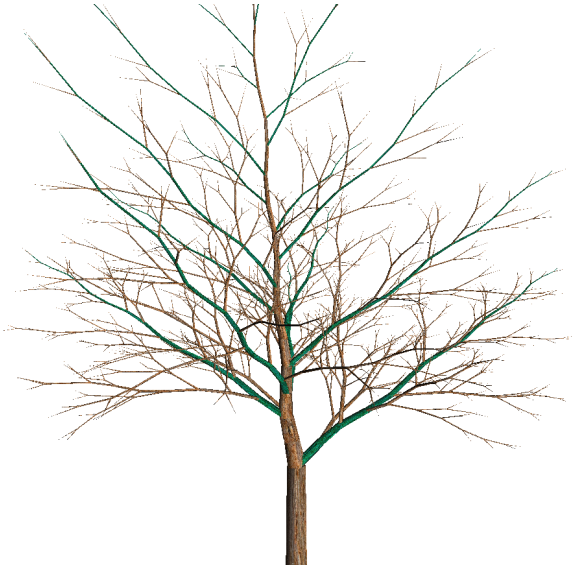


Slika 2.8: Interaktivno načrtovanje pomožnih parametrov α^g in α^w za izračun kotov vejitev $\alpha^{g,w}$.

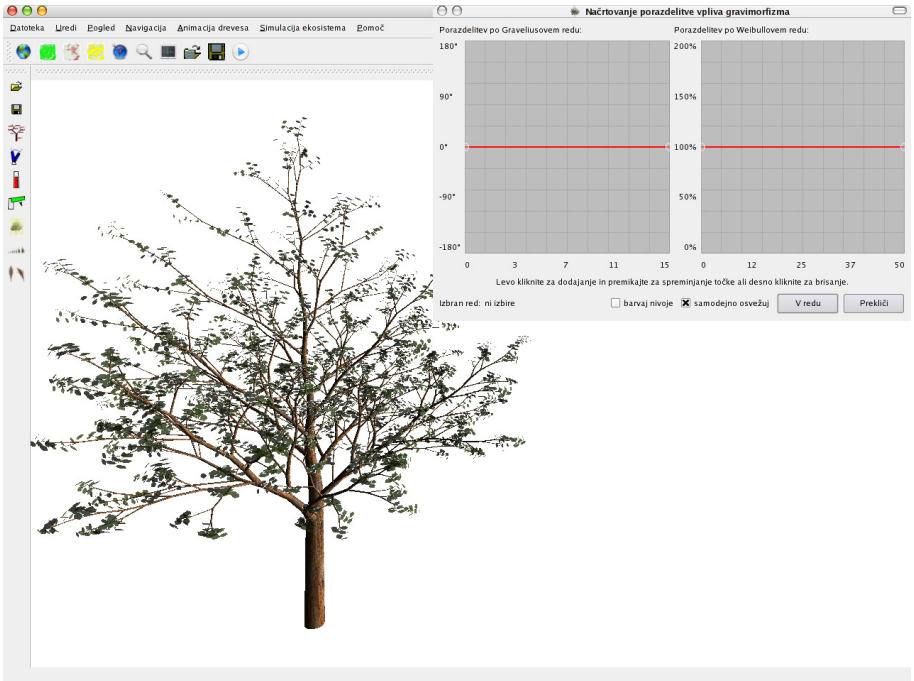
S premikanjem krmilnih točk določimo potek vrednosti parametra na ordinatni osi v odvisnosti od Graveliusovega in Weibullovega reda (na abscisni osi). Čeprav sta oba reda celoštevilski vrednosti, položajev krmilnih točk nismo omejili na abscisni osi, ker bi s tem zmanjšali okretnost modeliranja. Rezultat modeliranja je, da dobimo določene lokalne parametre za vse celoštevilske vrednosti redov g in w. V dodatno pomoč pri oblikovanju krmilne lomljenke parametra je vizualni namig, pri katerem se del drevesne strukture, na katerega bo vplivala sprememba izbrane krmilne točke oz. odseka lomljenke, pobarva z drugačno barvo (slika 2.9, tu so obarvane vse veje z Graveliusovim redom $g = 1$ z zeleno barvo). Vsaka sprememba oblike krmilne lomljenke se takoj odraža na vizualiziranem modelu, zato je mogoče v relativno kratkem času doseči skoraj poljubno drevesno obliko (slike 2.10, 2.11, 2.12, 2.13). V pogovornem oknu na slikah smo spreminjali grafa pomožnih parametrov za parameter gravimorfizma $\alpha_{\mathrm{m}}^{g,w}$.

Gradnja geometrijske strukture dreves

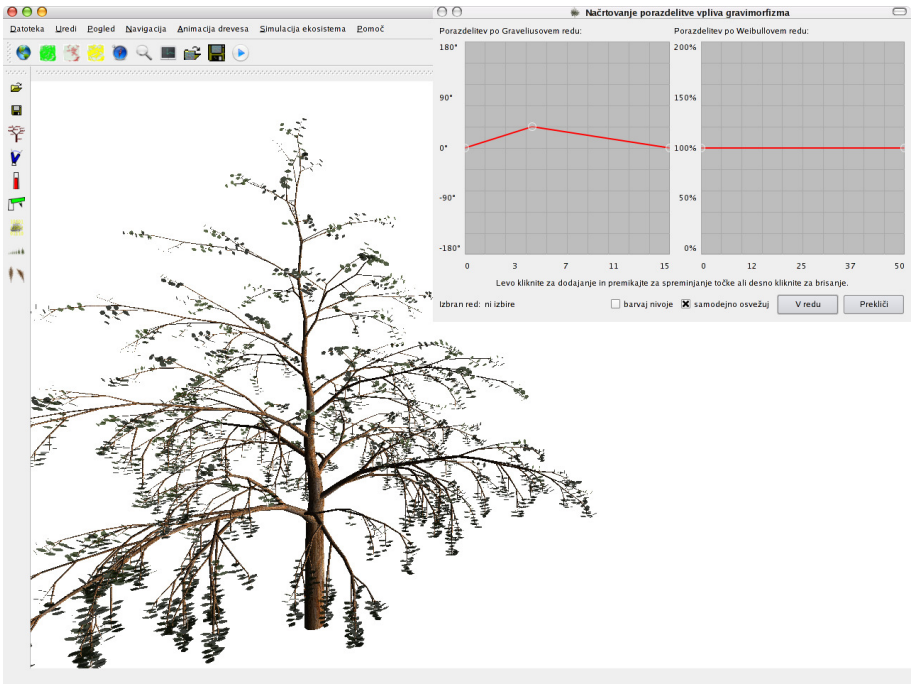
Modelirnik na osnovi podanih oz. z grafi oblikovanih vrednosti parametrov zgradi žilni model in sproti tvori geometrijsko strukturo drevesa. Medtem ko Holton kot geometrijske gradnike uporablja ploskve NURBS, smo mi deblo in veje predstavili kot zaporedje



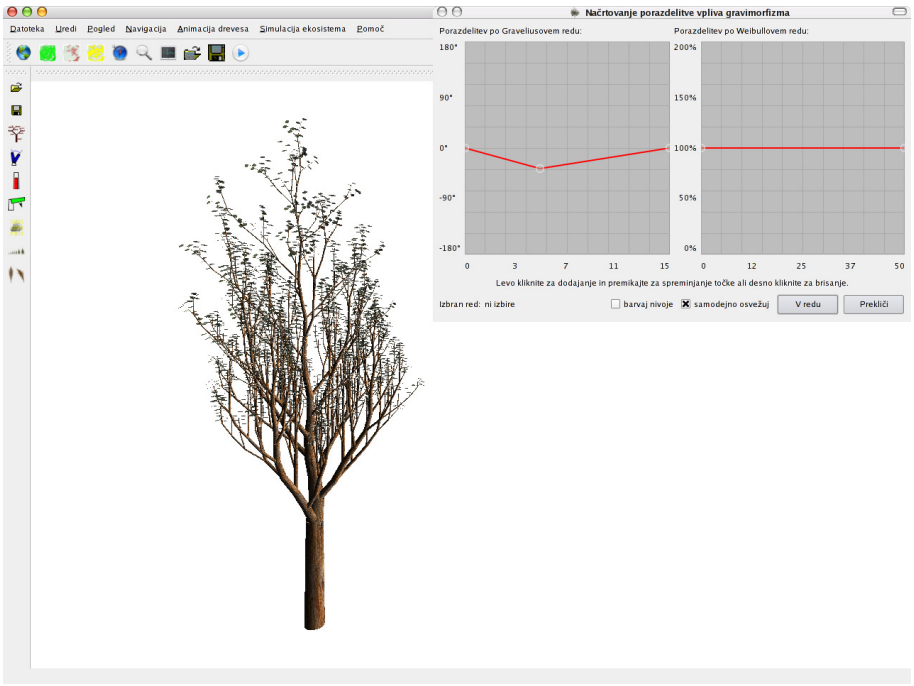
Slika 2.9: Interaktivno obarvanje delov veje, na katere bo vplivalo spreminjanje lokalnega parametra $\alpha^{g,w}$.



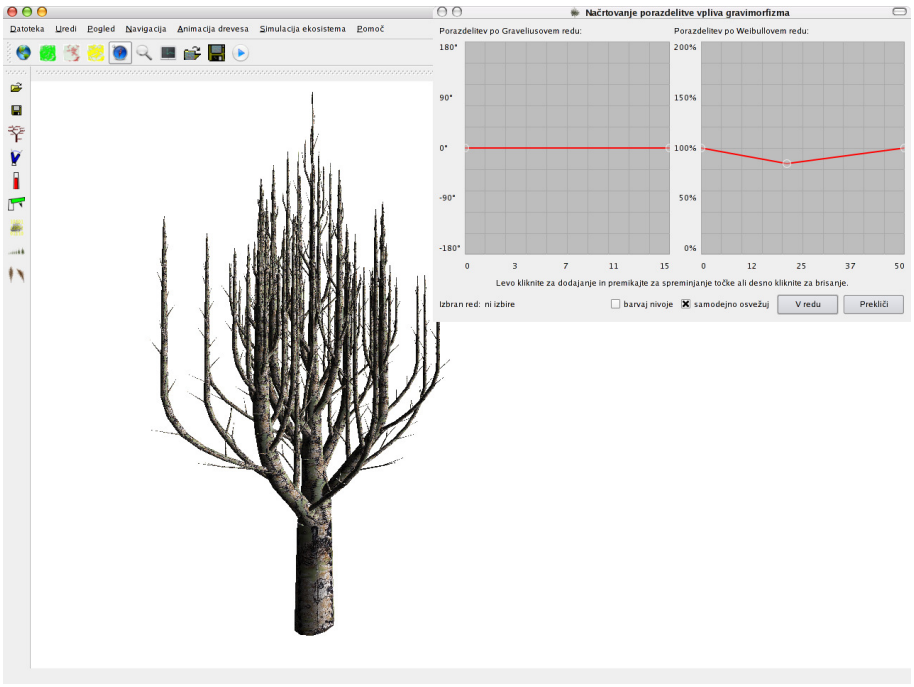
Slika 2.10: Interaktivno oblikovanje modela drevesa, osnovno drevo.



Slika 2.11: Interaktivno oblikovanje modela drevesa, dodan pozitivni gravimorfizem.



Slika 2.12: Interaktivno oblikovanje modela drevesa, dodan negativni gravimorfizem po redu g.



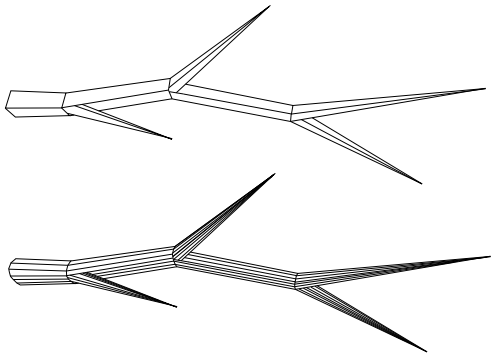
Slika 2.13: Interaktivno oblikovanje modela drevesa, dodan negativni gravimorfizem po redu w.

povezanih prisekanih piramid (slika 2.14), podobno kot v modelu Weber-Penn. Razlog za to je v hitrosti in z njo povezani zmožnosti interaktivne obdelave ter veliko enostavnejši animaciji. Listi oz. iglice so štirikotniki s teksturo, ki lahko vsebuje prozorna območja in tako določa poljubno obliko lista. V nadaljevanju bomo na kratko opisali razlike v matematičnem modelu za gradnjo geometrijske strukture drevesa glede na Holtonov model.

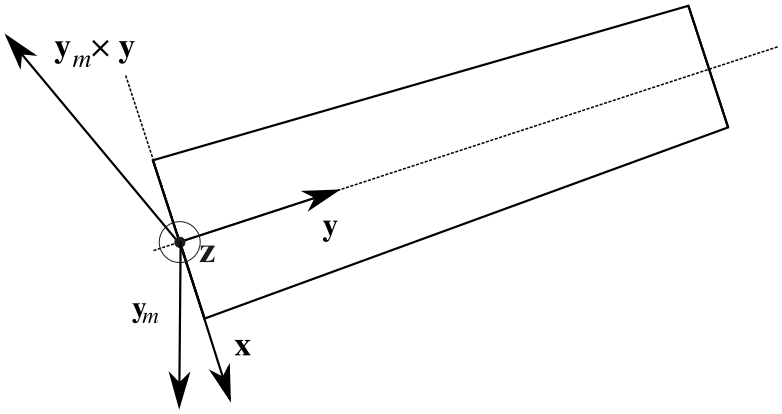
Gravimorfizem smo definirali s kotom $\alpha_{\mathrm{m}}^{g,w} \in [-180°, 180°]$ in s tem združili gravimorfizem s tendenco po rasti navzgor. Ker je vektor zasuka za gravimorfizem definiran v globalnem koordinatnem sistemu (t.j. prvi odsek debla), za ta koordinatni sistem vodimo inverzno matriko zasukov $\mathbf{M}_{\mathrm{m}}^{-1}$, ki je za glavno podvejo zmnožek naslednjih inverznih transformacij (matrik zasukov $\mathbf{R}$):

$$\mathbf{M}_{\mathrm{m};1}^{-1} = \mathbf{R}_{\mathbf{y}\times\mathbf{y}_{\mathrm{m}}}(-\alpha_{\mathrm{m}}^{g,w})\mathbf{R}_y(-\alpha_{\mathrm{p}})\mathbf{R}_z(-\alpha_1)\mathbf{M}_{\mathrm{m};0}^{-1}, \tag{2.63}$$

kjer je $\mathbf{y}_{\mathrm{m}}$ vektor iz globalnega koordinatnega sistema v smeri navzdol, izražen v koordinatnem sistemu trenutne veje in $\mathbf{y}$ vektor v smeri osi y v trenutnem koordinatnem sistemu. Z vektorskim produktom omenjenih vektorjev dobimo nanju pravokoten vektor $\mathbf{y} \times \mathbf{y}_{\mathrm{m}}$ (glej sliko 2.15), ki določa os zasuka veje. Če sučemo okoli tega dobljenega



Slika 2.14: Geometrijska struktura vej z vrsto prisekanih piramid. Zgoraj je prikazana struktura s 4-strano piramido, spodaj s 15-strano piramido. Notranje veje ponazorimo s prisekanimi piramidami, končne veje (veje z eno žilo) pa s piramidami.



Slika 2.15: Določitev vektorja zasuka pri gravimorfizmu. Skozi razvoj drevesne strukture si vodimo vektor $\mathbf{y}_{\mathrm{m}}$ v smeri navzdol, z vektorskim produktom tega vektorja in vektorja vzdolž veje v trenutnem koordinatnem sistemu, $\mathbf{y}$, pa dobimo nanju pravokoten vektor $\mathbf{y} \times \mathbf{y}_{\mathrm{m}}$.

vektorja, vejo upognemo navzdol glede na globalni koordinatni sistem. $\mathbf{M}^{-1}_{\mathrm{m};1}$ je inverzna matrika zasukov iz osnovne veje (glej sliko 2.27). Inverzna matrika $\mathbf{M}^{-1}_{\mathrm{m};0}$ ima na začetku vrednost identitete. Za stransko podvejo si analogno vodimo matriko $\mathbf{M}^{-1}_{\mathrm{m};2}$, pri čemer kot vejitve α_1 zamenjamo s kotom α_2.

Dolžino glavne podveje l_1 in stranske podveje l_2 do naslednje vejitve izračunamo:

$$r_1 = \max\left\{\min\left\{\sqrt{\frac{S_1}{S_0}}, M^{g,w}\right\}, m^{g,w}\right\}, \qquad r_2 = \max\left\{\min\left\{\sqrt{\frac{S_2}{S_0}}, M^{g,w}\right\}, m^{g,w}\right\}, \tag{2.64}$$

$$L_1 = r_1 L_0, \qquad L_2 = r_2 L_0, \tag{2.65}$$

$$l_1 = k_1^{g,w} L_1, \qquad l_2 = k_1^{g,w} L_2. \tag{2.66}$$

V enačbah nastopajo trije uporabniško nastavljivi parametri. Parametra $m^{g,w} \in [0, 20]$ in $M^{g,w} \in [0, 20]$ določata spodnjo in zgornjo mejo koeficientov r_1 ter r_2 in ju podamo

Slika 2.16: Skupina dreves, ki smo jo upodobili s senčilnikom iz programskega paketa LightWave.

s pomočjo krmilnih lomljenk, kot prikazuje slika 2.8. r_1 določa relativno dolžino glavne podveje, L_1, glede na dolžino osnovne veje L_0, medtem ko r_2 določa relativno dolžino stranske podveje, L_2. Parameter $k_1^{g,w}$ je skalirni faktor, s pomočjo katerega izračunamo dejansko dolžino glavne podveje, l_1, in dolžino stranske podveje, l_2.

Iz geometrijske strukture drevesa tvorimo trodimenzionalni model tako, da veje oblečemo s prisekanimi stožci. Prisekane stožce aproksimiramo z n-stranimi prisekanimi piramidami ($n \in \{4, 15\}$, slika 2.14).

Geometrijsko strukturo drevesa lahko izrišemo takoj ali shranimo posamezna oglišča štirikotnikov. Točke dobimo tako, da njihove koordinate izrazimo v koordinatnem sistemu prvega odseka debla, kar dobimo iz matrike M_0. Če se odločimo za shranjevanje točk, lahko te uvozimo v drugih programih, na primer LightWave (slika 2.16).

Modeliranje listov

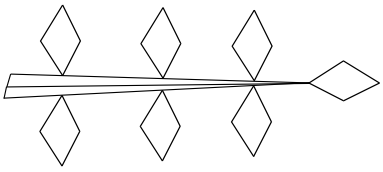
Listi so razporejeni enakomerno vzdolž končnih vej in rastejo pravokotno navzven od osi vzdolž veje. List upodobimo kot kvadrat, na katerega prilepimo teksturo. Slika za teksturo vsebuje informacije o transparentnosti, ki določi videno obliko lista. Primer dveh tekstur je viden na sliki 2.17. Ozadje teksture ima transparentnost, postavljeno na 1, zato je tisti del povsem prozoren.

Število in velikost listov na posamezni veji določata posebna parametra, še en parameter pa določa tip porazdelitve listov vzdolž veje. Tipi porazdelitev listov so določeni

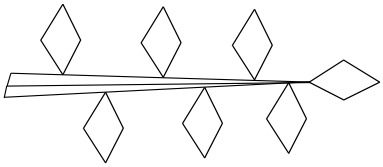
Slika 2.17: Primer dveh tekstur za liste.

po zgledu programskega paketa Corel Bryce 5 [232]. V kolikor je listov neparno število, je neparni list postavljen na konico veje. Ostali listi so razporejeni glede na enega od petih tipov:

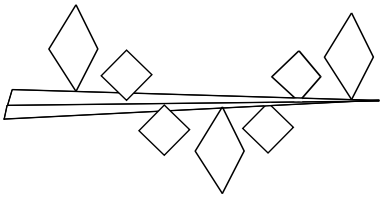
1. **nakopičeno** (ang. *stacked*) razporejeni listi so v paru po dva na vsaki strani veje, postavljeni pravokotno navzven od veje (slika 2.18),
2. **nestalno** (ang. *staggered*) razporejeni listi se izmenjujejo po eden na vsaki strani veje, postavljeni pravokotno na vejo (slika 2.19),
3. **spiralno** (ang. *spiral*) razporejeni listi se ovijajo okoli veje za nek določen kot pravokotno na vzdolžno os veje (slika 2.20),
4. **šopasto** (ang. *bunched*) razporejeni listi se zvrstijo vsi na koncu veje (slika 2.21) ali
5. **igličasto** (ang. *coniferous*) razporejeni listi so razporejeni kot nestalni in oklepajo z vejo ostri kot (slika 2.22).



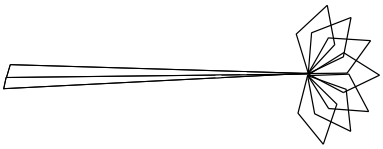
Slika 2.18: Nakopičeno razporejeni listi (tip 1).



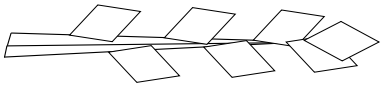
Slika 2.19: Nestalno razporejeni listi (tip 2).



Slika 2.20: Spiralno razporejeni listi (tip 3).



Slika 2.21: Šopasto razporejeni listi (tip 4).



Slika 2.22: Igličasto razporejeni listi (tip 5).

Poenostavljanje geometrijske strukture drevesa

V namen pohitritve izrisa je podprto tudi prilagajanje stopnje poenostavitve geometrije drevesa. Drevo iz gozda dreves označimo z indeksom (s, p), tedaj stopnjo podrobnosti l_{LOD} (*level-of-detail*, LOD) določimo iz oddaljenosti drevesa od gledišča $\mathbf{p}_v$ kot:

$$l_{\text{LOD}} = \left\lfloor \sqrt{\frac{\|\mathbf{p}_v - \mathbf{p}_{s,p}\|}{5}} \right\rfloor, \tag{2.67}$$

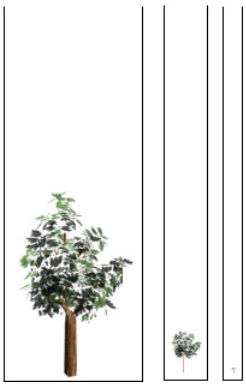
kjer je $\mathbf{p}_{s,p}$ lokacija drevesa z indeksom (s, p) in število 5 smo izkustveno izbrali po preskusu več vrednosti. Če je $l_{\text{LOD}} = 0$, poenostavljanja ne izvršimo. Sicer geometrijsko strukturo drevesa razvijamo le do vej z Weibullovim redom:

$$w_{\text{geom}} = (10 - l_{\text{LOD}})^2. \tag{2.68}$$

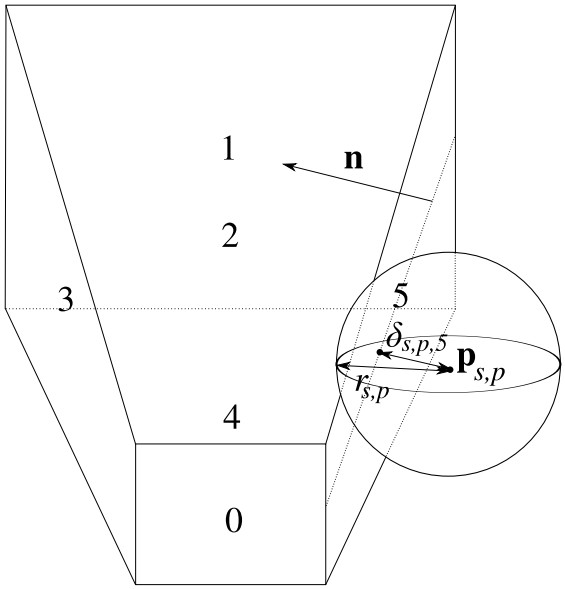
Nivoje zaključimo z listi, vendar nekaterih vej pri tem več ne rišemo. Veje rišemo do vej z Weibullovim redom:

$$w_{\text{branch}} = 10 - l_{\text{LOD}}. \tag{2.69}$$

Geometrijsko strukturo dreves poenostavimo glede oddaljenosti od opazovalca, tako da z izrisom prisekanih piramid pri bolj oddaljenih drevesih končamo pri manjšem redu w. Strukturo še gradimo, le piramid ne rišemo. Na sliki 2.23 vidimo isto drevo na različnih razdaljah od gledišča z izračunano stopnjo poenostavitve l_{LOD}. Drevo na levi nastane, ko je opazovalec 50 m stran ($l_{\text{LOD}} = 3$, veje do globine $w_{\text{branch}} = 7$, geometrijska struktura do največ $w_{\text{geom}} = 49$), drevo na sredini pri 200 m ($l_{\text{LOD}} = 6$, veje do globine $w_{\text{branch}} = 4$, geometrijska struktura do največ $w_{\text{geom}} = 16$), drevo na desni pri 700 m ($l_{\text{LOD}} = 10$, samo deblo $w_{\text{branch}} = 0$, geometrijska struktura do največ $w_{\text{geom}} = 0$).



Slika 2.23: Poenostavljanje geometrijske strukture drevesa z oddaljenostjo od opazovalca.



Slika 2.24: Obrezovanje na vidni volumen.

Če je drevo v celoti izven vidnega volumna, drevesa ne narišemo. Tako na sceni z 10.000 drevesi, kjer imamo v vidnem volumnu le 100 dreves, izrišemo le-te, kar precej poveča zmogljivost aplikacije. Da ugotovimo, ali bo drevo prikazano, moramo testirati, ali je kak del geometrijske očrtajoče krogle drevesa znotraj vidnega volumna. Hitra rešitev je testiranje, na kateri strani ravnin vidnega volumna se objekt nahaja [179].

Naša naloga je ugotoviti, ali je obkrožajoča krogla znotraj vidnega volumna, zunaj ali pa če ga obkroža. Rešitev dobimo na naslednji način: za vseh 6 ravnin vidnega volumna testiramo, ali je krogla na notranji strani ravnine, zunanji strani ravnine ali pa ravnino seka. Na sliki 2.24 s primerjavo radija $r_{s,p}$ obkrožajoče krogle drevesa s centrom v $\mathbf{p}_{s,p}$ in oddaljenostjo od ravnine $\delta_{s,p,5}$ določimo, na kateri strani ravnine 5 se drevo nahaja. Če ravnino predstavimo z eksplicitno enačbo:

$$ax + by + cz + d = 0 \tag{2.70}$$

in jo delimo s $\sqrt{a^2 + b^2 + c^2}$, da dobimo:

$$Ax + By + Cz + D = 0, \tag{2.71}$$

tedaj je vektor $\overline{\mathbf{n}} = [A, B, C]^T$ normaliziran vektor normale in D razdalja vzdolž $\overline{\mathbf{n}}$ od te ravnine do koordinatnega izhodišča. Normalo postavimo tako, da kaže notri proti vidnemu volumnu. Razdaljo δ od poljubne točke $\mathbf{p}_{s,p} = [x_{s,p} \;\; y_{s,p} \;\; z_{s,p}]^T$ do te ravnine izračunamo po enačbi:

$$\delta = \mathbf{p}_{s,p}\overline{\mathbf{n}} + D. \tag{2.72}$$

Da bi ugotovili, ali se obkrožajoča krogla drevesa seka z ravnino, izračunamo razdaljo δ od središča krogle $\mathbf{p}_{s,p}$ do ravnine. Razdalja δ je lahko pozitivna ali negativna in to določa, na kateri strani ravnine točka $\mathbf{p}_{s,p}$ leži. Če je razdalja manjša od radija, je drevo zunaj vidnega volumna. Če je absolutna vrednost manjša od radija, se drevo seka z ravnino, sicer nadaljujemo s testiranjem še za preostale ravnine. Opisan postopek podaja algoritem 12.

Algoritem 12 Izračun vsebnosti drevesa v vidnem volumnu.

Vhod: $\mathbf{p}_{s,p}$ - središče obkrožajoče krogle drevesa; $r_{s,p}$ - radij obkrožajoče krogle; D_i - razdalja od izhodišča do ravnine vzdolž normale ravnine.

Izhod: tip vsebnosti drevesa v vidnem volumnu.

1: **for** $i = 0$ **to** 5 **do**
2: $\quad\delta_{s,p,i} = \mathbf{p}_{s,p}\overline{\mathbf{n}}_i + D_i$; {skalarni produkt z lokacijo drevesa in razdalja do ravnine}
3: $\quad$**if** $\delta_{s,p,i} < -r_{s,p}$ **then**
4: $\quad\quad$**return** OUT; {$\delta_{s,p,i} + r_{s,p} < 0$, drevesa ne narišemo}
5: $\quad$**end if**
6: $\quad$**if** $|\delta_{s,p,i}| < r_{s,p}$ **then**
7: $\quad\quad$**return** $INTERSECT$; {$\delta_{s,p,i} - r_{s,p} < 0$ ali $\delta_{s,p,i} + r_{s,p} < 0$, drevo narišemo}
8: $\quad$**end if**
9: **end for**
10: **return** IN; {drevo obkroža vidni volumen, ga narišemo}

Za ta postopek potrebujemo enačbe ravnin, ki si odvisne od pogleda v OpenGL. Enačbe ravnin pri uporabi OpenGL hitro in direktno dobimo iz matrike pogleda [86].

Animacija rasti drevesa

Podprli smo dve obliki animacije dreves, simulacijo rasti in zibanje drevesa v vetru. Simulacijo rasti dosežemo z večanjem začetnega števila žil v drevesu. Če želimo doseči rast drevesa, linearno odvisno od časa, moramo število žil S večati kvadratično s časom t (to pomeni, da se debelina drevesa z letnicami linearno povečuje). Hkrati povečujemo dejansko dolžino prvega odseka debla $l_0^{0,0}$, ki določa višino, na kateri se pojavijo prve veje (s tem linearno povečujemo dolžine vseh vej). Če čas rasti rastline t normaliziramo, število žil pri animaciji S' in dejansko dolžino prvega odseka debla ($l_0^{0,0'}$) določimo kot:

$$S' = t^2 S, \tag{2.73}$$

$$l_0^{0,0'} = t^2 l_0^{0,0}. \tag{2.74}$$

Primer rasti drevesa prikazuje slika 2.25. Prvo drevo na sliki levo ima 100 žil, zadnje 1000. Normalizirana starost zadnjega drevesa je $t = 1$, prvega pa $t = 0{,}01$.

Slika 2.25: Simulacija rasti drevesa.

Animacija gibanja v vetru

Animirali smo dva tipa gibanja v vetru, prvi je pozibavanje v vetru spremenljive smeri, drugi je nagib vej v smeri stalnega (usmerjenega) vetra. Parametri drevesa, ki vplivajo na izračunani model gibanja, so prožnost vej k_{f}, dejanska dolžina veje l_0, moč vetra spremenljive smeri w_{s}, smer usmerjenega vetra $\mathbf{w}$ in moč usmerjenega vetra w_{g}.

Pri animaciji pihanja neusmerjenega vetra vejo zasučemo okoli osi vzdolž veje in osi pravokotne na os vzdolž veje (slika 2.26). Kota omenjenih zasukov označimo z α_x in α_z ter izračunamo glede na čas t po enačbah:

$$\alpha_x(t) = w_{\mathrm{s}}(1 - k_{\mathrm{f}})l_0 \sin(t + R_x), \qquad \alpha_z(t) = w_{\mathrm{s}}(1 - k_{\mathrm{f}})l_0 \sin(t + R_z), \tag{2.75}$$

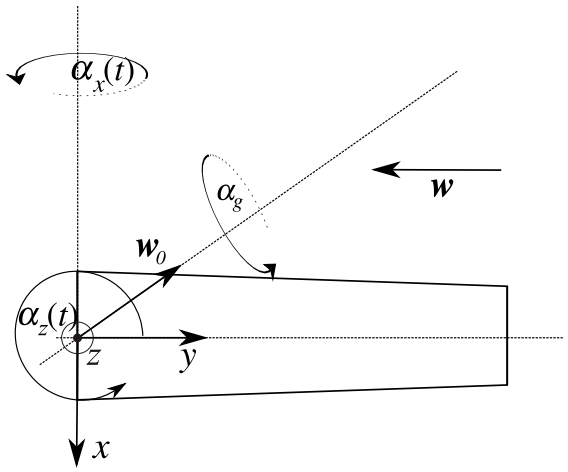
pri čemer sta R_x in R_z psevdonaključni racionalni števili za vsako vejo ($R_x, R_y \in [0, 2\pi]$). Tako se veje pri animaciji gibanja v vetru na videz zibajo neodvisno ena od druge. Na tak način je animacija precej bolj realistična, saj lahko v nasprotnem primeru hitro opazimo ponavljajoči se vzorec valovanja pri zibanju vej.

Pri animaciji usmerjenega vetra normaliziran vektor $\mathbf{w}_0$ (smer vetra) določa smer zasuka veje. Jakost usmerjenega vetra w_{g} določa kot zasuka α_g okoli vektorja $\mathbf{w}$ (slika 2.26), ki je odvisen od števila žil S_0:

$$\alpha_w = \frac{S_0}{S} w_{\mathrm{g}}. \tag{2.76}$$

Vektor $\mathbf{w}_0$ izračunamo kot:

$$\mathbf{w}_0 = \mathbf{y} \times \mathbf{M}_{\mathbf{w}}^{-1}\mathbf{w}, \tag{2.77}$$



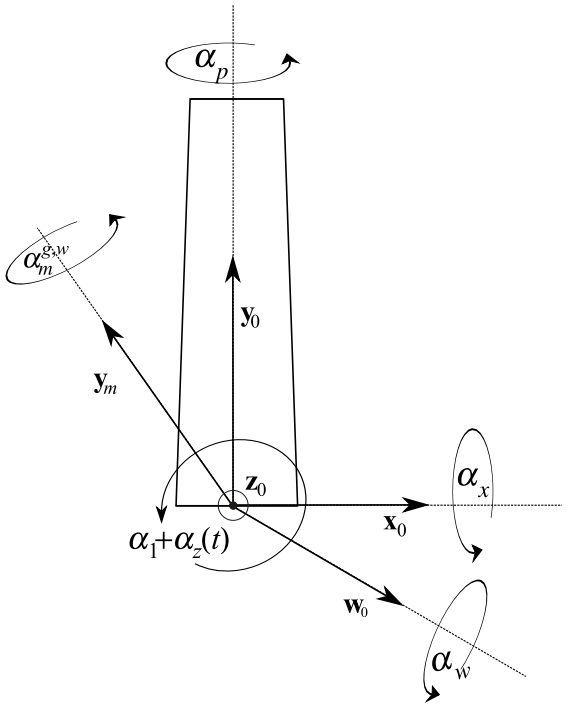
Slika 2.26: Zasuk veje zaradi usmerjenega vetra.

kjer $\mathbf{y}$ označuje vektor v koordinatnem sistemu trenutne veje v smeri y in matrika $\mathbf{M}_{\mathbf{w}}^{-1}$ določa inverzno transformacijo koordinatnega sistema trenutne veje v koordinatni sistem prvega odseka debla oz. globalni koordinatni sistem, v katerem je določen vektor $\mathbf{w}$. Inverzno matriko $\mathbf{M}_{\mathbf{w}}^{-1}$ zaradi hitrejšega izračunavanja vodimo sproti za vsako opravljeno transformacijo, tako da nad obstoječo matriko izvedemo inverzno transformacijo. Matriko $\mathbf{M}_{\mathbf{w};1}^{-1}$ za glavno podvejo dobimo iz matrike $\mathbf{M}_{\mathbf{w};0}^{-1}$ v osnovni podveji po naslednji enačbi:

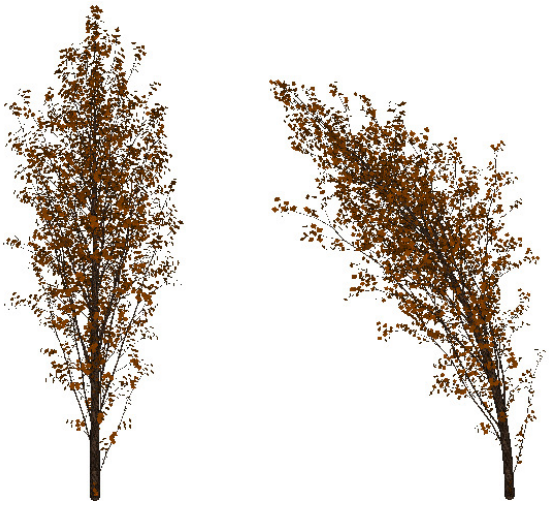
$$\mathbf{M}_{\mathbf{w};1}^{-1} = \mathbf{R}_{\mathbf{y}\times\mathbf{y}_\mathrm{m}}(-\alpha_\mathrm{m}^{g,w})\mathbf{R}_y(-\alpha_\mathrm{p})\mathbf{R}_x(-\alpha_x(t))\mathbf{R}_z(-\alpha_1-\alpha_z(t))\mathbf{R}_{\mathbf{w}_0}(-\alpha_w)\mathbf{M}_{\mathbf{w};0}^{-1}, \quad (2.78)$$

kjer vektor $\mathbf{w}_0$ označuje vektor v smeri vetra iz koordinatenega sistema prvega odseka debla, izražen v trenutnem koordinatnem sistemu. Navedeni zasuki v algoritmu so vidni na sliki 2.27. Vektorja $\mathbf{x}_0$ in $\mathbf{y}_0$ označujeta vektorja v smeri x in y, izražena v koordinatnem sistemu trenutne veje. $\mathbf{R}$ označuje matriko zasuka okoli ustrezne smeri (osi, vektorja) za podan kot. Učinek vetra na drevo vidimo na sliki 2.28, kjer je na levi drevo brez učinka vetra, na desni je učinek vetra $w_\mathrm{s} = 1,5$ in $w_\mathrm{g} = 2,5$. Po opravljenih zasukih se pomaknemo vzdolž veje in rekurzivno nadaljujemo gradnjo drevesa.

Celotnen postopek za izračun naše geometrijske strukture in animacije drevesa podaja algoritem 13. Kot vidimo iz algoritma, v vsakem rekurzivnem klicu prejmemo nabor lokalnih in globalnih parametrov. V izračunu enega rekurzivnega klica najprej izračunamo debelino trenutne osnovne veje in jo upodobimo z n-strano piramido. V kolikor smo na veji z eno žilo, ustrezno razporedimo nanjo še liste. Nato nadaljujemo z izračunom lastnosti obeh podvej in transformiramo koordinatni matriki obeh podvej



Slika 2.27: Zasuki veje pri razvijanju geometrijske strukture drevesa.



Slika 2.28: Učinek vetra na geometrijo drevesa.

Algoritem 13 Izračun geometrijske strukture drevesa. Rekurzivno proceduro pokličemo z **drevo**(0, 0, S, 1, $l_0^{0,0}$, **I**, **I**, **I**).

Vhod: g, w - Graveliusov in Weibullov indeks osnovne veje; S_0 - število žil v osnovni veji; L_0 - relativna dolžina osnovne veje; l_0 - dejanska dolžina osnovne veje; $\mathbf{M}_0$ - koordinatni sistem trenutne veje; $\mathbf{M}_{\mathrm{m};0}^{-1}$ - inverzna matrika zasukov za gravimorfizem v koordinatnem sistemu osnovne veje; $\mathbf{M}_{\mathbf{w};0}^{-1}$ - inverzna matrika zasukov za usmerjen veter v koordinatnem sistemu osnovne veje; globalni k_d, k_c, l_type, $k_\mathrm{s}^{g,w}$, $M^{g,w}$, $m^{g,w}$, $k_\mathrm{l}^{g,w}$, $\alpha_\mathrm{m}^{g,w}$, $\alpha^{g,w}$,t, k_f, w_s, w_g (str. 49).

Izhod: upodobljeno drevo

postopek drevo(g, w, S_0, L_0, l_0, $\mathbf{M}_0$, $\mathbf{M}_{\mathrm{m};0}^{-1}$, $\mathbf{M}_{\mathbf{w};0}^{-1}$):

$d := k_\mathrm{d}\sqrt{S_0}$; {izračun debeline osnovne veje}

upodobi osnovno vejo ($\mathbf{M}_0$, l_0, d);

if $S_0 = 1$ **then**

  upodobi liste(l_type);

  return;

end if

$S_1 := \lceil 1 + k_\mathrm{s}^{g,w}\,(S_0 - 2)\rceil$; {število žil v glavni podveji}

$S_2 = S_0 - S_1$; {število žil v stranski podveji}

$r_1 := \max\left\{\min\left\{\sqrt{\frac{S_1}{S_0}}, M^{g,w}\right\}, m^{g,w}\right\}$; {določimo razmerje dolžin}

$r_2 := \max\left\{\min\left\{\sqrt{\frac{S_2}{S_0}}, M^{g,w}\right\}, m^{g,w}\right\}$;

$L_1 := r_1 L_0$; {relativna dolžina podvej}

$L_2 := r_2 L_0$;

$l_1 := k_\mathrm{l}^{g,w} L_1$; {določimo dejansko dolžina podvej}

$l_2 := k_\mathrm{l}^{g,w} L_2$;

$\alpha_1 := k_\mathrm{c}\sqrt{\frac{S_2}{S_0}}\alpha^{g,w}$; {izračunamo kote vejitev}

$\alpha_2 := \alpha^{g,w} - \alpha_1$;

$\alpha_x(t) := \sin(t + R_x) w_\mathrm{s}(1 - k_\mathrm{f}) l_0$; {animacija neusmerjenega vetra}

$\alpha_z(t) := \sin(t + R_z) w_\mathrm{s}(1 - k_\mathrm{f}) l_0$;

$\alpha_w := \frac{S_0}{S} w_\mathrm{g}$; {animacija usmerjenega vetra}

$\mathbf{M}_1 := \mathbf{R}_{\mathbf{w}_0}(\alpha_w,)\mathbf{R}_z(\alpha_1 + \alpha_z(t))\mathbf{R}_x(\alpha_x(t))\mathbf{R}_y(\alpha_\mathrm{p})\mathbf{R}_{\mathbf{y}\times\mathbf{y}_\mathrm{m}}(\alpha_\mathrm{m}^{g,w})\mathbf{T}_y(l_0)\mathbf{M}_0$;

$\mathbf{M}_2 := \mathbf{R}_{\mathbf{w}_0}(\alpha_w,)\mathbf{R}_z(\alpha_2 + \alpha_z(t))\mathbf{R}_x(\alpha_x(t))\mathbf{R}_y(\alpha_\mathrm{p})\mathbf{R}_{\mathbf{y}\times\mathbf{y}_\mathrm{m}}(\alpha_\mathrm{m}^{g,w})\mathbf{T}_y(l_0)\mathbf{M}_0$;

$\mathbf{M}_{\mathrm{m};1}^{-1} := \mathbf{R}_{\mathbf{y}\times\mathbf{y}_\mathrm{m}}(-\alpha_\mathrm{m}^{g,w})\mathbf{R}_y(-\alpha_\mathrm{p})\mathbf{R}_x(-\alpha_x(t))\mathbf{R}_z(-\alpha_1 - \alpha_z(t))\mathbf{M}_{\mathrm{m};0}^{-1}$;

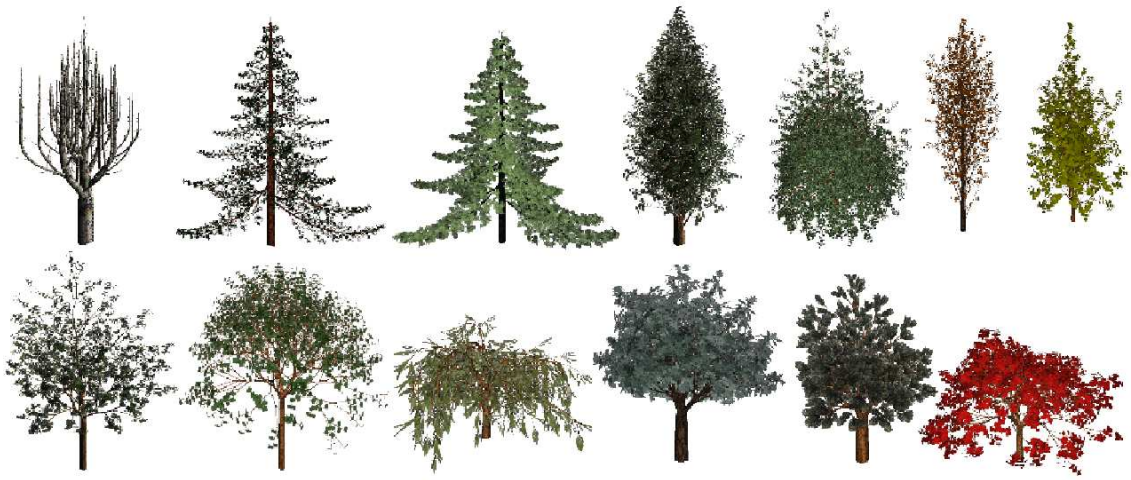
$\mathbf{M}_{\mathrm{m};2}^{-1} := \mathbf{R}_{\mathbf{y}\times\mathbf{y}_\mathrm{m}}(-\alpha_\mathrm{m}^{g,w})\mathbf{R}_y(-\alpha_\mathrm{p})\mathbf{R}_x(-\alpha_x(t))\mathbf{R}_z(-\alpha_2 - \alpha_z(t))\mathbf{M}_{\mathrm{m};0}^{-1}$;

$\mathbf{M}_{\mathbf{w};1}^{-1} := \mathbf{R}_{\mathbf{y}\times\mathbf{y}_\mathrm{m}}(-\alpha_\mathrm{m}^{g,w})\mathbf{R}_y(-\alpha_\mathrm{p})\mathbf{R}_x(-\alpha_x(t))\mathbf{R}_z(-\alpha_1 - \alpha_z(t))\mathbf{R}_{\mathbf{w}_0}(-\alpha_w)\mathbf{M}_{\mathbf{w};0}^{-1}$;

$\mathbf{M}_{\mathbf{w};2}^{-1} := \mathbf{R}_{\mathbf{y}\times\mathbf{y}_\mathrm{m}}(-\alpha_\mathrm{m}^{g,w})\mathbf{R}_y(-\alpha_\mathrm{p})\mathbf{R}_x(-\alpha_x(t))\mathbf{R}_z(-\alpha_2 - \alpha_z(t))\mathbf{R}_{\mathbf{w}_0}(-\alpha_w)\mathbf{M}_{\mathbf{w};0}^{-1}$;

drevo ($g + 1$, $w + 1$, S_2, L_2, l_2, $\mathbf{M}_2$, $\mathbf{M}_{\mathrm{m};2}^{-1}$, $\mathbf{M}_{\mathbf{w};2}^{-1}$); {razvijemo stransko vejo}

drevo (g, $w + 1$, S_1, L_1, l_1, $\mathbf{M}_1$, $\mathbf{M}_{\mathrm{m};1}^{-1}$, $\mathbf{M}_{\mathbf{w};1}^{-1}$); {razvijemo glavno vejo}



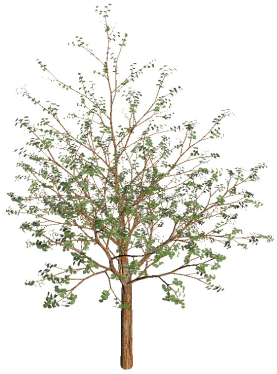
Slika 2.29: Primeri upodobitev proceduralnih modelov dreves s sistemom EcoMod.

ter inverzne matrike za vektorja gravimorfizma in vektorja usmerjenega vetra v koordinatnih sistemih obeh podvej. Rotacijo zibanja v vetru apliciramo tik pred izrisom veje, po apliciranju ostalih rotacij. Vektor v smeri sunka vetra vodimo ločeno od vektorja v smeri začetka rasti debla, ki je potrebnen za apliciranje gravimorfizma, saj se vektor gravitacije v vzdolžni smeri začetka rasti drevesa ne spreminja zaradi pihanja vetra, temveč je stalen v razvoju drevesa, ki raste iz tal.

2.2.4 Primeri modeliranja z našim modelom

Drevo lahko modeliramo interaktivno ali parametre vpišemo v vhodno datoteko. Globalni parametri so vpisani v datoteko en za drugim v nespremenjeni obliki. Grafi vektorskih parametrov so v datoteko zapisani uniformno, z normaliziranimi vrednostmi koordinat točk krmilne lomljenke. Interaktivno globalne parametre nastavimo z drsniki ali vpišemo numerične vrednosti, tip listov pa izberemo iz spustnega menija (glej sliko 2.7 na strani 51). Sliki za teksture izberemo z dialogom za izbiro datotek. Matrične parametre z indeksi (g, w) nastavljamo z dvema lomljenkama, kot je opisano na strani 52, v podrazdelku 2.2.3. Aplikacija omogoča tudi izvoz parametriziranega proceduralnega modela drevesa s kodiranjem računskih formul v obliki LaTeX [130], ki jim doda vrednosti interaktivno parametriziranih parametrov.

Nekaj primerov upodobitev proceduralnih modelov dreves s sistemom EcoMod vidimo na sliki 2.29. Kot vidimo, je izbran proceduralni model dovolj fleksibilen, da je možno upodobiti tako igličasta, kot listnata drevesa, z različnimi vejitvenimi strukturami, barvami listov in drugimi posebnimi značilnostmi.



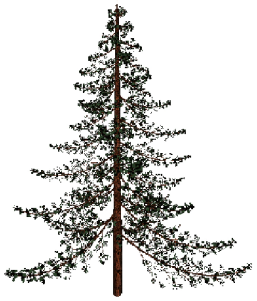
Slika 2.30: Upodobitev proceduralnega modela za bukev.

Bukev

Za modeliranje bukve iz slike 2.30 smo izbrali naslednje globalne parametre: $S = 1000$, $l_0^{0,0} = 5$, $k_\mathrm{d} = 0{,}01$, $\alpha_\mathrm{p} = 85$, $k_\mathrm{c} = 0{,}3$, $l_\mathrm{l} = 0{,}1$, $\rho_\mathrm{l} = 5$, $l_\mathrm{type} =$ `Spiral` in $l_\mathrm{LOD} = 0$. Vrednosti k_s^g in k_s^w za razmerje porazdelitve žil $k_\mathrm{s}^{g,w}$ pri vseh redih veje (g, w) dobimo iz grafov lomljenk s krmilnimi točkami $k_\mathrm{s}^g = \{(0,\ 0{,}75),\ (15,\ 0{,}9)\}$ in $k_\mathrm{s}^w = \{(0,\ 1),\ (50,\ 1)\}$. Za vrednosti α^g in α^w za kote med izhajajočimi podvejami pri delitvah, $\alpha^{g,w}$, definiramo krmilne točke $\alpha^g = \{(0,\ 45°),\ (15,\ 45°)\}$ in $\alpha^w = \{(0,\ 1),\ (50,\ 1)\}$. Spodnjo mejo relativne dolžine veje, $m^{g,w}$, dobimo z uporabo $m^g = \{(0,\ 1),\ (7{,}5,\ 0{,}45),\ (15,\ 1)\}$ in $m^w = \{(0,\ 1),\ (50,\ 1)\}$. Zgornjo mejo relativne dolžine veje, $M^{g,w}$, dobimo podobno $M^g = \{(0,\ 1),\ (15,\ 1)\}$ in $M^w = \{(0,\ 1),\ (50,\ 1)\}$. Skalirni faktor dolžine veje, $k_\mathrm{l}^{g,w}$, za to drevo izračunamo s pomočjo $k_\mathrm{l}^g = \{(0,\ 1),\ (15,\ 1)\}$ in $k_\mathrm{l}^w = \{(0,\ 1),\ (50,\ 1)\}$. Kot zasuka za gravimorfizem, $\alpha_\mathrm{m}^{g,w}$, določimo s pomočjo $\alpha_\mathrm{m}^g = \{(0,\ 0°),\ (15,\ 0°)\}$ in $\alpha_\mathrm{m}^w = \{(0,\ 1),\ (50,\ 1)\}$.

Smreka

Za modeliranje smreke na sliki 2.31 smo izbrali naslednje globalne parametre: $S = 6000$, $l_0^{0,0} = 5{,}47$, $k_\mathrm{d} = 0{,}005$, $\alpha_\mathrm{p} = 85$, $k_\mathrm{c} = 0{,}98$, $l_\mathrm{l} = 0{,}1815$, $\rho_\mathrm{l} = 30$, $l_\mathrm{type} =$ `Stacked` in $l_\mathrm{LOD} = 0$. Vrednosti k_s^g in k_s^w smo določili $k_\mathrm{s}^g = \{(0,\ 0{,}955),\ (0{,}7749,\ 0{,}9615),\ (1{,}827,\ 0{,}8288),\ (4{,}207,\ 0{,}8269),\ (15,\ 0{,}8294)\}$ in $k_\mathrm{s}^w = \{(0,\ 1),\ (42{,}89,\ 1{,}005),\ (42{,}89,\ 0{,}654),\ (50,\ 0{,}6256)\}$. Pomožne parametre kotov vejitev definirata grafa z lomljenkama $\alpha^g = \{(0,\ 128{,}8°),\ (1{,}164,\ 17{,}91°),\ (3{,}621,\ 45{,}21°),\ (15,\ 35{,}83°)\}$ in $\alpha^w = \{(0,\ 1),\ (50,\ 1)\}$. Spodnjo mejo relativne dolžine veje, $m^{g,w}$, parametriziramo z $m^g = \{(0,\ 0{,}7109),\ (1{,}293,\ 0{,}8531),\ (15,\ 1)\}$ in $m^w = \{(0,\ 1),\ (50,\ 1)\}$. Zgornjo mejo relativne dolžine veje, $M^{g,w}$, parametriziramo z $M^g = \{(0,\ 0{,}9953),\ (15,\ 1)\}$



Slika 2.31: Upodobitev proceduralnega modela za smreko.

in $M^w = \{(0,\ 1),\ (50,\ 1)\}$. Skalirni faktor dolžine veje, $k_{\mathrm{l}}^{g,w}$, parametriziramo s $k_{\mathrm{l}}^g = \{(0,\ 0{,}7109),\ (0{,}7749,\ 0{,}5385),\ (2{,}522,\ 1{,}327),\ (15,\ 1)\}$ in $k_{\mathrm{l}}^w = \{(0,\ 1{,}014),\ (38{,}58,\ 1{,}014),\ (45{,}26,\ 0{,}5877),\ (50,\ 0{,}01896)\}$. Kot zasuka za gravimorfizem, $\alpha_{\mathrm{m}}^{g,w}$, parametriziramo z $\alpha_{\mathrm{m}}^g = \{(0,\ 0°),\ (0{,}3875,\ 34{,}62°),\ (1{,}034,\ -4{,}265°),\ (1{,}55,\ -4{,}154°),\ (2{,}974,\ -5{,}972°),\ (15,\ -0{,}8532°)\}$ in $\alpha_{\mathrm{m}}^w = \{(0,\ 1),\ (50,\ 1)\}$.

Vrba žalujka

Za modeliranje vrbe iz slike 2.32 smo izbrali naslednje globalne parametre: $S = 1500$, $l_0^{0,0} = 1$, $k_{\mathrm{d}} = 0{,}01$, $\alpha_{\mathrm{p}} = 85$, $k_{\mathrm{c}} = 0{,}2$, $l_{\mathrm{l}} = 0{,}3452$, $\rho_{\mathrm{l}} = 5$, $l_{\mathrm{type}} =$ `Coniferous` in $l_{\mathrm{LOD}} = 0$. Razmerje porazdelitve žil $k_{\mathrm{s}}^{g,w}$ določimo preko vrednosti k_{s}^g in k_{s}^w z grafoma podanima kot lomljenki $k_{\mathrm{s}}^g = \{(0,\ 0{,}8507),\ (3{,}75,\ 0{,}7749),\ (15,\ 0{,}9)\}$ in $k_{\mathrm{s}}^w = \{(0,\ 1),\ (50,\ 1)\}$. Kot med izhajajočima podvejama pri delitvi, $\alpha^{g,w}$ določimo posredno preko $\alpha^g = \{(0,\ 17{,}91°),\ (1{,}875,\ 57{,}16°),\ (15,\ 45°)\}$ in $\alpha^w = \{(0,\ 1),\ (50,\ 1)\}$. Spodnjo mejo relativne dolžine veje, $m^{g,w}$, določimo s parametri $m^g = \{(0,\ 1),\ (1{,}81,\ 0{,}9953),\ (15,\ 1)\}$ in $m^w = \{(0,\ 0{,}9668),\ (50,\ 1)\}$. Zgornjo mejo relativne dolžine veje, $M^{g,w}$, določimo s pomočjo $M^g = \{(0,\ 0{,}9953),\ (1{,}422,\ 1{,}848),\ (15,\ 1)\}$ in $M^w = \{(0,\ 1),\ (50,\ 1)\}$. Skalirni faktor dolžine veje, $k_{\mathrm{l}}^{g,w}$, določimo s $k_{\mathrm{l}}^g = \{(0,\ 1),\ (15,\ 1)\}$ in $k_{\mathrm{l}}^w = \{(0,\ 1),\ (22{,}63,\ 0{,}9953),\ (37{,}07,\ 0{,}7678),\ (50,\ 1)\}$. Kot zasuka za gravimorfizem,

Slika 2.32: Upodobitev proceduralnega modela za vrbo (zunanji upodabljalnik).

$\alpha_{\mathrm{m}}^{g,w}$, določimo z $\alpha_{\mathrm{m}}^{g} = \{(0,\ 84,45°), (1{,}746,\ 94{,}69°), (15,\ 110°)\}$ in $\alpha_{\mathrm{m}}^{w} = \{(0,\ 0{,}6446), (50,\ 1)\}$. Slika 2.32 je upodobljena v zunanjem upodabljalniku preko modula za uvoz geometrije.

Kodirne specifike modelirnika EcoMod

Aplikacija EcoMod je kodirana v programskem jeziku C++ in uporablja Qt4 in OpenGL, zato je neodvisna od operacijskega sistema (Linux/Mac/Win32, ...), procesorske arhitekture (32/64 bitna) in prevajalnika C++ (GCC, MinGW, MSVC). Sestavljajo jo trije glavni medsebojno povezani moduli, izmed katerih ima vsak poseben namen:

1. Modelirnik geometrije 3D naravnih dreves je prva, precej interaktivna komponenta aplikacije. Proceduralni model drevesa temelji na opisanem Holtonovem žilnem modelu, ki je še dopolnjen in spremenjen na več načinov, npr. za lažjo parametrizacijo ali več animacijskih tehnik. Omogoča interaktivno parametrizacijo proceduralnega modela drevesa z uporabo grafov in drugih interaktivnih gradnikov. Dobljeni parametri se shranijo v datoteko. Takšne datoteke je možno kasneje naložiti, z aplikacijo pa prihaja tudi osnovna knjižnica primerov zmodeliranih dreves. Upodobitev proceduralnega modela je opravljena z OpenGL, okenski kontekst za OpenGL pa je pridobljen s Qt4 gradnikom QGLWidget.

2. Vizualizator ekosistemov je druga komponenta, ki upodablja realno pokrajino realnega terena. Za prikaz površja so uporabljena polja oglišč. Programska koda uporablja knjižnico GLee za delo z OpenGL razširitvami za platformno neodvisnost le-teh. Drevesni modeli so dobljeni iz komponente za 3D modeliranje ter nameščeni na mesta rasti dreves iz simulacije rasti ekosistema. Scena lahko vsebuje do več sto tisoč dreves, t.j. do skupaj več kot 100.000.000 trikotnikov. Izdelani primerki proceduralnih modelov so zato za pohitritev prikazovanja poenostavljeni glede na stopnjo podrobnosti ali odstranjeni zaradi obrezovanja na vidni volumen.

3. Simulacija umetnega življenja dreves oz. rastlin znotraj ekosistema uporablja realne podatke življenjskih pogojev, tekmovanje med rastlinami in model razširjenja rastlin. Stohastična simulacija uporablja LCG generator naključnih števil. Realni podatki so dobljeni iz digitalnega modela reliefa (podatki iz sistemov GIS), vlažnost, vetrovnost in osončenost naloženega terena pa izračunani sproti. Ta komponenta je lahko uporabljena kot ogrodje, za uporabo številnih njenih algoritmov in modelov v drugih, multi-disciplinarnih področjih.

2.3 Avtomatsko modeliranje z rekonstrukcijo in slikovni pristopi

Modeliranje s slikovnimi pristopi zajema uporabo slik, npr. dvodimenzionalnih fotografij, iz katerih želimo izluščiti značilke in jih uporabiti v svojem geometrijskem modelu [190]. Z rekonstrukcijo drevesom podobnih, vendar enostavnejših, struktur se srečamo v medicini [38, 75], inovacije z rekonstrukcijskimi postopki pa so vse bolj komercialno uporabne s porastom vseprisotnega računalništva in razširjene resničnosti [190].

Poskus uporabe rekonstrukcijskega postopka pri načrtovanju morfološke strukture dreves, s pomočjo skiciranja dvodimenzionalne topologije na projekcijsko sliko, najdemo že v pristopu [193], ki sešteva volumetrične gostote. V pristopu [40] uporabnik poda topologijo osnovnih vej drevesa, algoritem pa rekurzivno deterministično vstavi samopodobno nadaljnjo rast vej. V [178] so načrtovali fiziologijo dreves, da bi ocenili prostornino drevesne krošnje.

Hyyppa s sod. [106] razdeli aplikativnost trodimenzionalnih odsevov površja s sistemi LiDAR (ang. *Light Detection And Ranging*) na področje dreves in gozdov v naslednje kategorije:

- lociranje na terenu in določitev višine dreves,
- izločitev značilk (porazdelitev višine dreves in posamezne značilnosti dreves),
- razpoznava drevesnih vrst in letnega prirastka ter
- uporabo drugih spektralnih tehnik za določitev gozdnih značilnosti.

Von Mammen in Jacob [234] naključne prepisne nize generirata z L-sistemi s pomočjo paralelnega mehanizma rojenja delcev [120, 24] in ob upodabljanju nudita uporabniku določeno možnost interakcije v obliki gojenja, ki je za simulacijo ekosistemov vključen tudi v našem sistemu EcoMod [254].

Beaumont in Stepney [12] sta rekonstruirala dvodimenzionalna drevesa s predstavitvijo v L-sistemih. Napredek rekonstrukcije sta merila s podobnostjo slike referenčnega drevesa in slike upodobljenega parametriziranega proceduralnega modela. Njuna metoda obsega zgolj dvodimenzionalni poskus rekonstrukcije in je precej časovno zahtevna. Drevesni modeli, ki sta jih dobila, so zgolj dvodimenzionalne projekcije vej v obliki povezanih črt z nespremenljivimi koti vejitve na istih nivojih. Referenčne slike, ki sta jih uporabila za primerjavo, so generirane iz kratkih prepisnih pravil s kontekstno neodvisnimi L-sistemi.

3. Diferencialna evolucija za rekonstrukcijo parametriziranih proceduralnih drevesnih modelov

Domišljija je pomembnejša od znanja. (Albert Einstein)

Da bi potrdili prvi dve zastavljeni hipotezi v okviru teze doktorske disertacije, pokažimo, kako je možno združiti optimizacijski algoritem diferencialne evolucije (opisan v podpoglavju 2.1) in numerično kodiran proceduralni model dreves (opisan v podpoglavju 2.2). Predlagan rekonstrukcijski postopek kodira parametre proceduralnega modela drevesa iz ogrodja EcoMod [268] v genotipu posameznika iz populacije algoritma jDE [23]. S predlaganim postopkom rekonstruiramo trodimenzionalni proceduralni model drevesa, tako da z algoritmom jDE skušamo ustrezno parametrizirati parametre proceduralnega modela [270, 257]. Rekonstrukcijski postopek temelji na primerjavi slik dreves, ki predstavljajo dvodimenzionalne projekcije referenčnih trodimenzionalnih dreves. Rekonstrukcija je toliko bolj uspešna, kolikor so si slike referenčnih dreves in slike upodobljenih parametriziranih proceduralnih modelov podobne. Parametrizacija proceduralnega modela poteka postopno in, bolj, kot so si slike podobne, boljša (t.j. manjša) je optimirana vrednost ovrednotenja primerjave razlik med drevesi [266, 267].

V našem pristopu bomo za predstavitev trodimenzionalnega drevesa uporabili svoj zgrajen proceduralni model [271, 268], ki ga skušamo z rekonstrukcijo ustrezno parametrizirati. Rekonstrukcijski postopek temelji na razpoznavi dvodimenzionalnih fotografij naravnih dreves. Rekonstrukcija je toliko bolj uspešna, kolikor so si slike fotografij dreves in slike upodobljenih parametriziranih proceduralnih modelov podobne [12, 190]. Parametrizacija proceduralnega modela poteka postopno [233, 234], z evolucijo po vzoru iz narave, za kar uporabimo enega od svetovno najboljših evolucijskih algoritmov, algoritem jDE [23, 29, 266, 24].

V nadaljevanju opisujemo nekatere pomembnejše dele rekonstrukcijskega postopka, t.j. kodiranje genotipa, preslikavo genotipa v fenotip in ovrednotenje fenotipa preko primerjave z referenčno sliko.

3.1 Kodiranje genotipa

Za predstavitev drevesa smo uporabili svoj numerično kodiran proceduralni drevesni model iz programskega sistema EcoMod [268], predstavljenega v podpoglavju 2.2.3. Ta model uporabniku pomaga zgraditi drevesno geometrijo, tako da zmanjša nabor parametrov, ki jih uporabnik mora nastaviti, da bi modeliral geometrijski model drevesa. Tak nabor parametrov sestavljajo npr. debeline vej, relativne dolžine vej in proporci vejitvene strukture. Nabor poimenujemo tudi gojitveno seme drevesa [233, 234], saj so v njem zajeti vsi parametri za izračun drevesne geometrije. Ko namreč uporabnik poda to seme, postopek v proceduralnem modelu samodejno rekurzivno določi drevesno strukturo (t.j. morfologijo) in s tem lokacije, rotacije, velikosti ter teksture sestavnih delov drevesa, kot so več tisoč vejnih odsekov in več tisoč listov. Vsak korak rekurzije izračuna dodaten drevesni gradnik v geometrijskem modelu. Na tak način nad danim naborom numerično kodiranih vhodnih parametrov, kot so npr. širina debla, relativna dolžina vej in vejitvena struktura, procedura zgradi trodimenzionalno drevesno strukturo [249]. Končni geometrijski model posameznega drevesa je generiran s parametrizacijo proceduralnega modela. Parametriziran proceduralni model lahko kasneje uporabimo za računalniško animacijo, saj s spremembo glavnih ali pomožnih parametrov modela dobimo nekoliko spremenjene podobne modele, ki so primerni za izračunavanje ključnih animacijskih okvirjev [269].

Kot smo definirali v podpoglavju 2.2.3, parametre izbranega proceduralnega modela ločimo na skalarne in matrične (vektorske). Ta nabor skalarnih in matričnih parametrov tvori v prejšnjem odstavku omenjeno gojitveno seme. V [268] smo predstavili tudi vizualno interaktivno orodje s posebno parametrizacijo matričnih parametrov s konvolucijo parametrov po indeksih, ki smo jih definirali z lomljenko na grafu. Izbran model se od nekaterih drugih modelov [5, 15, 96, 195, 167] razlikuje v tem, da je popolnoma numerično kodiran in je nespremenljive dimenzije. Zaradi tega je ta model še posebej primeren za iskanje parametrov z uporabo diferencialne evolucije [266].

Posamezen genotipni vektor $\mathbf{x}$, $i \in \{0, ..., NP-1\}$ iz populacije v algoritmu jDE kodira nabor parametrov proceduralnega modela v gojitvenem semenu. Dimenzionalnost razvijanega realno-kodiranega genotipa $\mathbf{x}$ je $D = 4509$. Genotip sestoji iz komponent $\mathbf{x} = x_{i,j} \in [0, 1]$, $j \in \{0, ..., D-1\}$, ki kodirajo naslednje parametre gojitvenega semena:

- število žil v drevesu $S = 400x_{i,0} + 10$, $S \in [10, 410]$ (določa kompleksnost drevesa: enako številu vej z listi, vejnih odsekov manj ena ali polovici vej manj dve),

- višino prvega odseka debla $l_0^{0,0} = 10x_{i,1}$, $l_0^{0,0} \in [0\text{ m}, 10\text{ m}]$,
- koeficient debeline veje $k_\mathrm{d} = 0{,}05x_{i,2}$, $k_\mathrm{d} \in [0,\ 0{,}05]$,
- kot filotakse $\alpha_\mathrm{p} = 360x_{i,3}$, $\alpha_\mathrm{p} \in [0°, 360°]$,
- razmerje porazdelitve žil na podveji pri vejitvah $k_\mathrm{s}^{g,w} = 0{,}5x_{i,j} + 0{,}5$, $\forall j \in \{4, ...,753\}$, $k_\mathrm{s}^{g,w} \in [0{,}5,\ 1]$,
- kot med izhajajočima podvejama pri delitvi $\alpha^{g,w} = 180x_{i,j}$, $\forall j \in \{754, ..., 1503\}$, $\alpha^{g,w} \in [0°, 180°]$,
- zgornja meja relativne dolžine podvej glede na dolžino osnovne veje $M^{g,w} = 20x_{i,j}$, $\forall j \in \{1504, ..., 2253\}$, $M^{g,w} \in [0, 20]$,
- spodnja meja relativne dolžine podvej glede na dolžino osnovne veje $m^{g,w} = 20x_{i,j}$, $\forall j \in \{2254, ..., 3003\}$, $m^{g,w} \in [0, 20]$,
- skalirni faktor dolžine veje $k_\mathrm{l}^{g,w} = 20x_{i,j}$, $\forall j \in \{3004, ..., 3753\}$, $k_\mathrm{l}^{g,w} \in [0, 20]$,
- gravicentralizem $k_\mathrm{c} = x_{i,3754}$, $k_\mathrm{c} \in [0, 1]$,
- gravimorfizem (t.j. upogibanje vej zaradi gravitacije) $\alpha_\mathrm{m}^{g,w} = 360x_{i,j} - 180$, $\forall j \in \{3755, ..., 4504\}$, $\alpha_\mathrm{m}^{g,w} \in [-180°, 180°]$,
- vključitev listov na drevesu $B_\mathrm{l} = x_{i,4505} < 0{,}5\ ?\ 0 : 1$, $B_\mathrm{l} \in \{0, 1\}$,
- velikost listov $l_\mathrm{l} = 0{,}3x_{i,4506}$, $l_\mathrm{l} \in [0,\ 0{,}3]$ in
- gostoto listov $\rho_\mathrm{l} = 30x_{i,4507}$, $\rho_\mathrm{l} \in [0, 30]$,
- tip porazdelitve listov $l_\mathrm{type} = 5\lfloor x_{i,4508} \rfloor$ z indeksiranimi vrednostmi `Spiral`, `Stacked`, `Staggered`, `Bunched` in `Coniferous`,

kjer je $g \in [0, 15]$, $w \in [0, 50]$ in po 750 realnih parametrov kodira po eno matriko matričnega parametra. Pri tem j označuje indeks komponente v genotipu in za matrične parametre v gojitvenem semenu teče kot $j = o + (50g + w)$, kjer z o označimo indeksni odmik začetka kodiranja vektorskega parametra iz gojitvenega semena v genotipu.

Z opisanim kodiranjem je možno predstaviti številne različne proceduralne modele dreves [268], podajmo nekaj primerov. Drevo, podobno bukvi, ki je prikazano kot četrto drevo iz leve na sliki 3.1, nastane iz gojitvenega semena s $S = 400$ žilami in

drugimi skalarnimi parametri $S = 400$, $L_0 = 5$, $k_\mathrm{d} = 0{,}01$, $\alpha_\mathrm{p} = 85$, $k_\mathrm{c} = 0{,}3$, $l_\mathrm{l} = 0{,}1$, $\rho_\mathrm{l} = 5$, $B_\mathrm{l} = 1$, $l_\mathrm{type} =$ `Spiral`, $w_\mathrm{g} = 1{,}5$, $\mathbf{w} = [0\ 1\ 0]^T$, $k_\mathrm{f} = 1$ ter $l_\mathrm{LOD} = 0$. Da lažje opišemo še matrične parametre tega semena, jih bomo podali s pomočjo pomožnih enodimenzionalnih vektorskih paramerov, ki preko konvolucije svojih grafov posredno izražajo pripadajoče matrične parametre. Grafe, ki jih nad temi vektorji definiramo, podamo z lomljenkami, kot smo omenili zgoraj. Definicija teh grafov lahko v našem orodju EcoMod poteka tudi interaktivno, z določitvijo krmilnih točk njihovih lomljenk [268]. Vrednosti na ordinatni osi so realna števila, ki jih na položajih indeksov g in w vzorčimo iz interpolirane lomljenke. Matrični parameter porazdelitve žil tako izračunamo iz pomožnih enodimenzionalnih vektorskih parametrov iz enačbe 2.57. Za omenjeno drevo iz slike 3.1 pomožna parametra $k_\mathrm{s}^g \in [\frac{1}{2}, 1]$ in $k_\mathrm{s}^w \in [0, 2]$ zavzameta vrednosti $\mathbf{k}_s^g = \{(0,\ 0,75),\ (15,\ 0,9)\}$ in $\mathbf{k}_s^w = \{(0,1),\ (50,1)\}$. Matrični parameter kotov vejitev med izhajajočima podvejama izračunamo iz enačbe 2.58 in za omenjeno drevo iz slike 3.1 pomožna parametra $\alpha^g \in [0°, 180°]$ in $\alpha^w \in [0, 2]$ zavzameta vrednosti $\boldsymbol{\alpha}^g = \{(0, 45°),\ (15, 45°)\}$ in $\boldsymbol{\alpha}^w = \{(0,1),\ (50,1)\}$. Matrični parameter največje relativne dolžine vej izračunamo iz enačbe 2.60 in za omenjeno drevo iz slike 3.1 pomožna parametra $M^g \in [0, 10]$ in $M^w \in [0, 2]$ zavzameta vrednosti $\mathbf{M}^g = \{(0,1),\ (15,1)\}$ in $\mathbf{M}^w = \{(0,1),\ (50,1)\}$. Matrični parameter najmanjše relativne dolžine vej izračunamo iz enačbe 2.61 in za omenjeno drevo iz slike 3.1 pomožna parametra $m^g \in [0, 10]$ in $m^w \in [0, 2]$ zavzameta vrednosti $\mathbf{m}^g = \{(0,1),\ (7,5,\ 0,45),\ (15,1)\}$ in $\mathbf{m}^w = \{(0,1),\ (50,1)\}$. Matrični parameter skaliranja dolžine vej izračunamo iz enačbe 2.62 in za omenjeno drevo iz slike 3.1 pomožna parametra $k_\mathrm{l}^g \in [0, 10]$ in $k_\mathrm{l}^w \in [0, 2]$ zavzameta vrednosti $\mathbf{k}_\mathrm{l}^g = \{(0,1),\ (15,1)\}$ in $\mathbf{k}_\mathrm{l}^w = \{(0,1),\ (50,1)\}$. Matrični parameter vpliva gravitacije, gravimorfizem, izračunamo iz enačbe 2.59 in za omenjeno drevo iz slike 3.1 pomožna parametra $\alpha_\mathrm{m}^g \in [-180°, 180°]$ in $\alpha_\mathrm{m}^w \in [0, 2]$ zavzameta vrednosti $\boldsymbol{\alpha}_\mathrm{m}^g = \{(0, 0°),\ (15, 0°)\}$ in $\boldsymbol{\alpha}_\mathrm{m}^w = \{(0,1),\ (50,1)\}$. Gojitvena semena prvih štirih dreves iz leve na sliki 3.1 se razlikujejo zgolj v enem parametru, ki zaporedoma zavzame vrednosti $S = \{10, 50, 100, 400\}$. Gojitveno seme petega drevesa iz leve na tej sliki je enako kot pri četrtem, vendar je upodobljeno brez tekstur in listov ($B_\mathrm{l} = 0$), z žičnim modelom. Skrajno desno drevo se razlikuje glede na tretje v parametrih $k_\mathrm{s}^g = \{\ (0,\ 0{,}5),\ (15,\ 0{,}9)\ \}$.

Slika 3.1: Upodobitev proceduralnih modelov drevesa, podobnega bukvi, z različnim številom žil, žični model skeleta drevesne strukture brez listov in skrajno desno, enako drevo z drugačno vejitveno strukturo.

3.2 Preslikava genotipa v fenotip

Naš rekonstrukcijski postopek temelji na rekonstrukciji iz dvodimenzionalnih slik naravnih dreves $\mathbf{z}^* = \{z^*_{x,y}\}$, $\forall x = \{0, ..., X–1\}$, $\forall y = \{0, ..., Y–1\}$. Da bi z drevesom iz referenčnih slik $\mathbf{z}^*_i$ primerjali razvito tridimenzionalno drevo, kodirano v genotipu $\mathbf{x}$ dimenzije D, genotip najprej preslikamo v fenotip. Fenotip je nabor upodobljenih dvodimenzionalnih slik razvitega trodimenzionalnega drevesa, $\mathbf{z}_i = \{z^i_{x,y}\}$, $\forall x = \{0, ..., X\text{-}1\}$, $\forall y = \{0, ..., Y\text{-}1\}$. Fenotip izračunamo z grafičnim upodabljanjem genotipa $\mathbf{x}$, ki predstavlja seme za vhodne parametre parametriziranega proceduralnega modela za drevo v sistemu EcoMod. Geometrijski model in fenotipne slike izračunamo z algoritmom 14. Slednji algoritem pri gradnji sestavnih delov drevesa uporablja geometrijske interpolacije pri postavitvi krmilnih točk za vozlišča poligonov, ki jih projicira v teksturirane dvodimenzionalne barvne slike.

Slike $\mathbf{z}^*_i$ in $\mathbf{z}_i$ so dimenzij $X \times Y$. Slike pred medsebojno primerjavo še pretvorimo v črno bele, tako da vsak piksel materiala, ki je različen od ozadja, postavimo na 1 (črno), ostale piksle na 0 (belo). Slike referenčnega drevesa pred pričetkom optimizacije še skaliramo na dimenzijo slik projekcij razvitih modelov. Upodobitev istega razvitega parametriziranega proceduralnega modela vzorčimo večkrat v trodimenzionalni geometrijski model in nato dvodimenzionalno projekcijo, da dobimo različne projekcije. Vzorčimo vsaj dvakrat, iz različnih kotov pogleda kamere, npr. zamaknjenih za $\beta = 90°$ vzdolž debla. Vzrok za to je, da z izogibanjem ploskim drevesom zagotovimo trodimenzionalno razsežnost rekonstruiranih modelov. Če označimo algoritem 14 s funkcijskim predpisom $\mathbf{g}$, potem je $\mathbf{z}_i = \mathbf{g}(\mathbf{x}, \beta_c)$, $c = \{1, 2\}$ kjer $\beta_1 = 0°$ in $\beta_2 = 90°$.

3.3 Primerjava fenotipa z referenčno sliko

Fenotip ovrednotimo z oceno rekonstrukcijske napake. To oceno merimo z vsoto razlikovanj med slikami razvitih modelov in referenčnimi slikami. Rekonstrukcija je toliko bolj uspešna, kolikor bolj so istoležni piksli v referenčnih slikah dreves in v slikah upodobljenih parametriziranih proceduralnih modelov enaki. V ta namen smo izbrali naslednji kriterij. Metrika za kriterij je vsota manhattanskih razdalj od vsakega črnega (t.j. materialnega) piksla slike do najbližjega črnega piksla referenčne slike in obratno [12]. Posamezni fenotip tako ovrednotimo s kriterijem f_1:

$$f_1(\mathbf{x}) = f(\mathbf{g}(\mathbf{x}, \beta_1), \mathbf{g}(\mathbf{x}, \beta_2)) = h_1(\mathbf{z}_1) + h_1(\mathbf{z}_2) \tag{3.1}$$

$$h_1(\mathbf{z}_i) = \sum_{x,y} m_1(z^i_{x,y}, z^*_{i;x,y}) + \sum_{x,y} m_1(z^*_{i;x,y}, z^i_{x,y}) \tag{3.2}$$

kjer z m_1 označimo funkcijo za izračun manhattanske razdalje do najbližjega piksla v sliki $\mathbf{z}^*$ z vrednostjo 1 (t.j. črno, material), $i \in \{1, 2\}$. Kot vidimo, razlikovanje slik ovrednotenje ocene s povečanjem poslabša, zato je naloga optimizacije v našem primeru, najti čim manjše vrednosti obeh funkcij.

Definicijo celotnega predlaganega postopka *diferencialne evolucije za rekonstrukcijo parametriziranih proceduralnih modelov* podaja algoritem 15. Kot je vidno iz uporabe funkcije $\mathbf{g}$, algoritem z dvojno zanko `for` ovija algoritem 14, ki ocenjuje razvite modele. Po zaključku omenjenih dvojnih zank algoritem vrne nabor parametriziranih proceduralnih modelov, t.j. njihova gojitvena semena.

Algoritem 14 Izračun geometrijske strukture s proceduralnim drevesnim modelom. Rekurzivni algoritem pričnemo s klicem `odsekveje`$(0,\ 0,\ S,\ 1,\ l_0^{0,0},\ \mathbf{I},\ \mathbf{I})$, kjer $\mathbf{I}$ označuje matriko identitete.

Vhod: g, w - Graveliusov in Weibullov indeks osnovne veje; S_0 - skupno število žil v osnovni veji; L_0, l_0 - relativna in dejanska dolžina osnovne veje; $\mathbf{M}_0$ - koordinatni sistem osnovne veje; $\mathbf{M}_{\mathrm{m};0}^{-1}$ - v koordinatnem sistemu osnovne veje izražena inverzna rotacijska matrika za gravimorfizem; k_{d}, k_{c}, l_{type}, $k_{\mathrm{s}}^{g,w}$, $M^{g,w}$, $m^{g,w}$, $k_{\mathrm{l}}^{g,w}$, $\alpha_{\mathrm{m}}^{g,w}$, $\alpha^{g,w}$, t, k_{f}, w_{s}, w_{g} - podani parametri *semena*

Izhod: slika upodobljenega drevesa

postopek `odsekveje`$(g,\ w,\ S_0,\ L_0,\ l_0,\ \mathbf{M}_0,\ \mathbf{M}_{\mathrm{m};0}^{-1})$:
$d := k_{\mathrm{d}}\sqrt{S_0}$; {izračun debeline osnovne veje}
`upodobi osnovno vejo`$(\mathbf{M}_0,\ l_0,\ d)$;
if $S_0 = 1$ **then**
  `upodobi liste`(l_{type}); **return**; {zadnji nivo, upodobimo liste}
end if
$S_1 := \lceil 1 + k_{\mathrm{s}}^{g,w}\,(S_0 - 2)\rceil$, $S_2 = S_0 - S_1$; {število žil v podvejah}
$r_1 := \max\left\{\min\left\{\sqrt{\frac{S_1}{S_0}}, M^{g,w}\right\}, m^{g,w}\right\}$; {razmerja dolžin podvej glede na žile}
$r_2 := \max\left\{\min\left\{\sqrt{\frac{S_2}{S_0}}, M^{g,w}\right\}, m^{g,w}\right\}$;
$L_1 := r_1 L_0$, $L_2 := r_2 L_0$; {relativna dolžina podvej}
$l_1 := k_{\mathrm{l}}^{g,w} L_1$, $l_2 := k_{\mathrm{l}}^{g,w} L_2$; {aktivna dolžina podvej}
$\alpha_1 := k_{\mathrm{c}}\sqrt{\frac{S_2}{S_0}}\alpha^{g,w}$, $\alpha_2 := \alpha^{g,w} - \alpha_1$; {koti vejitve}
$\mathbf{M}_1 := \mathbf{R}_z(\alpha_1)\mathbf{R}_y(\alpha_{\mathrm{p}})\mathbf{R}_{\mathbf{y}\times\mathbf{y}_{\mathrm{m}}}(\alpha_{\mathrm{m}}^{g,w})\mathbf{T}_y(l_0)\mathbf{M}_0$; {matrike premika in rotacije}
$\mathbf{M}_2 := \mathbf{R}_z(\alpha_2)\mathbf{R}_y(\alpha_{\mathrm{p}})\mathbf{R}_{\mathbf{y}\times\mathbf{y}_{\mathrm{m}}}(\alpha_{\mathrm{m}}^{g,w})\mathbf{T}_y(l_0)\mathbf{M}_0$;
$\mathbf{M}_{\mathrm{m};1}^{-1} := \mathbf{R}_{\mathbf{y}\times\mathbf{y}_{\mathrm{m}}}(-\alpha_{\mathrm{m}}^{g,w})\mathbf{R}_y(-\alpha_{\mathrm{p}})\mathbf{R}_x(-\alpha_x(t))\mathbf{R}_z(-\alpha_1 - \alpha_z(t))\mathbf{M}_{\mathrm{m};0}^{-1}$; {osvežitev inverzne matrike za postavitev vektorja gravimorfizma}
$\mathbf{M}_{\mathrm{m};2}^{-1} := \mathbf{R}_{\mathbf{y}\times\mathbf{y}_{\mathrm{m}}}(-\alpha_{\mathrm{m}}^{g,w})\mathbf{R}_y(-\alpha_{\mathrm{p}})\mathbf{R}_x(-\alpha_x(t))\mathbf{R}_z(-\alpha_2 - \alpha_z(t))\mathbf{M}_{\mathrm{m};0}^{-1}$;
`odsekveje`$(g+1,\ w+1,\ S_2,\ L_2,\ l_2,\ \mathbf{M}_2,\ \mathbf{M}_{\mathrm{m};2}^{-1})$; {razvoj stranske veje}
`odsekveje`$(g,\ w+1,\ S_1,\ L_1,\ l_1,\ \mathbf{M}_1,\ \mathbf{M}_{\mathrm{m};1}^{-1})$; {razvoj glavne veje}
return; {sestop iz rekurzivnega izračuna za posamezen odsek veje v drevesu}

Algoritem 15 Rekonstrukcija parametriziranega proceduralnega modela drevesa iz slike.

postopek rekonstrukcija($\mathbf{z}^*$)

Vhod: S_0 - največje število žil v deblu; drugi parametri jDE [23] in EcoMod [268].

Izhod: rekonstruiran parametriziran proceduralni trodimenzionalni model drevesa.

`Uniformno naključno generiraj začetno populacijo DE` $\mathbf{x}_{i,0} \in [0,1]$ `za` $i = 1..NP$;

for DE generacijska zanka g (dokler FEs < 10000) **do**

  for DE iteracijska zanka i (za vse vektorje $\mathbf{x}_{i,g}$ v trenutni populaciji) **do**

    DE izračun poskusnega vektorja $\mathbf{x}_{i,g}$ (samoprilagajanje, mutacija, križanje):

$$F_{i,g+1} = \begin{cases} F_\mathrm{l} + rand_1 \times F_\mathrm{u} & \text{če } rand_2 < \tau_1, \\ F_{i,g} & \text{sicer} \end{cases};$$

$$CR_{i,g+1} = \begin{cases} rand_3 & \text{če } rand_4 < \tau_2, \\ CR_{i,g} & \text{sicer} \end{cases};$$

$$\mathbf{v}_{i,g+1} = \mathbf{x}_{r_1,g} + F_{i,g+1}(\mathbf{x}_{r_2,g} - \mathbf{x}_{r_3,g});$$

$$u_{i,j,g+1} = \begin{cases} v_{i,j,g+1} & \text{če } rand(0,1) \leq CR_{i,g+1} \text{ ali } j = j_\mathrm{rand} \\ x_{i,j,g} & \text{sicer} \end{cases};$$

    DE ocena ustreznosti (preslikava genotipa, upodabljanje in primerjava):

$\mathbf{z}_1 = \mathbf{g}(\mathbf{u}_{i,g}, \beta_1)$, $\mathbf{z}_2 = \mathbf{g}(\mathbf{u}_{i,g}, \beta_2)$ {Algoritem 14, dvakrat za en genotip}

$h(\mathbf{z}_1) = \sum_{x,y} m_1(z^1_{x,y}, z^*_{x,y}) + \sum_{x,y} m_1(z^*_{x,y}, z^1_{x,y})$; {Metrika podobnosti}

$h(\mathbf{z}_2) = \sum_{x,y} m_1(z^2_{x,y}, z^*_{x,y}) + \sum_{x,y} m_1(z^*_{x,y}, z^2_{x,y})$; {Za 3D, rotacija 90°}

$f(\mathbf{u}_{i,g}) = f(\mathbf{g}(\mathbf{u}_{i,g}, \beta_1), \mathbf{g}(\mathbf{u}_{i,g}, \beta_2)) = h(\mathbf{z}_1) + h(\mathbf{z}_2)$; {Ocena ustreznosti}

    DE selekcija:

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g+1} & \text{če } f(\mathbf{u}_{i,g+1}) < f(\mathbf{x}_{i,g}) \\ \mathbf{x}_{i,g} & \text{sicer} \end{cases};$$

  end for

end for

vrni najboljšega najdenega posameznika;

4. Rezultati

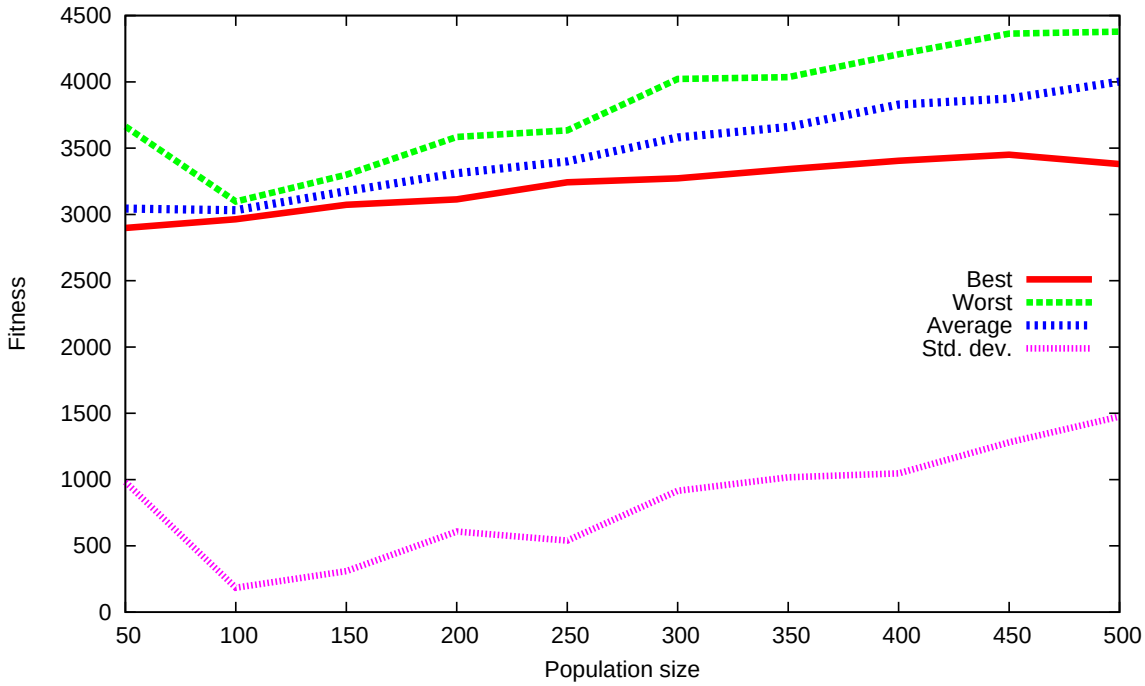
42. (K)

V tem poglavju prikažemo preskus predstavljenega rekonstrukcijskega algoritma. V nadaljevanju podajamo nastavitev parametrov za preskuse, dobljene rezultate in podajamo razpravo o njih.

4.1 Nastavitev parametrov

Algoritem našega pristopa za rekonstrukcijo dreves smo ocenili s preskusom na testu s primerom drevesa, vidnega skrajno desno na sliki 4.1. Največje število ovrednotenj kriterijske funkcije (FEs) za algoritem jDE smo določili kot 10000, kar je enako pristopu v [12]. Kot referenčno sliko smo primer iz evolucijskega zagona iz preliminarnega testiranja. Za dimenzijo vzorčenja upodobljenega parametriziranega proceduralnega modela smo izbrali 250x250. Največje število žil v drevesu smo omejili na $S = 410$, kar precej presega izraznostno kompleksnost vejitvene strukture v sorodnem pristopu [12], ki za predstavitev vejitvene strukture izvaja le štiri iteracije nad dokaj enostavnim prepisnim pravilom in konstantnim ravninskim vejitvenim kotom (cca. $S \approx 30$). Ostale parametre v jDE in sistemu EcoMod smo pustili enake privzetim vrednostim originalnih algoritmov iz literature.

Slika 4.1: Upodobljeni razviti parametrizirani proceduralni modeli pri $FEs = \{1, 8, 18, 1992, 2727, 3230\}$ in skrajno desno, referenčna slika.



Slika 4.2: Ocene zmogljivosti algoritma jDE, odvisne od velikosti populacije.

4.2 Dobljeni rezultati

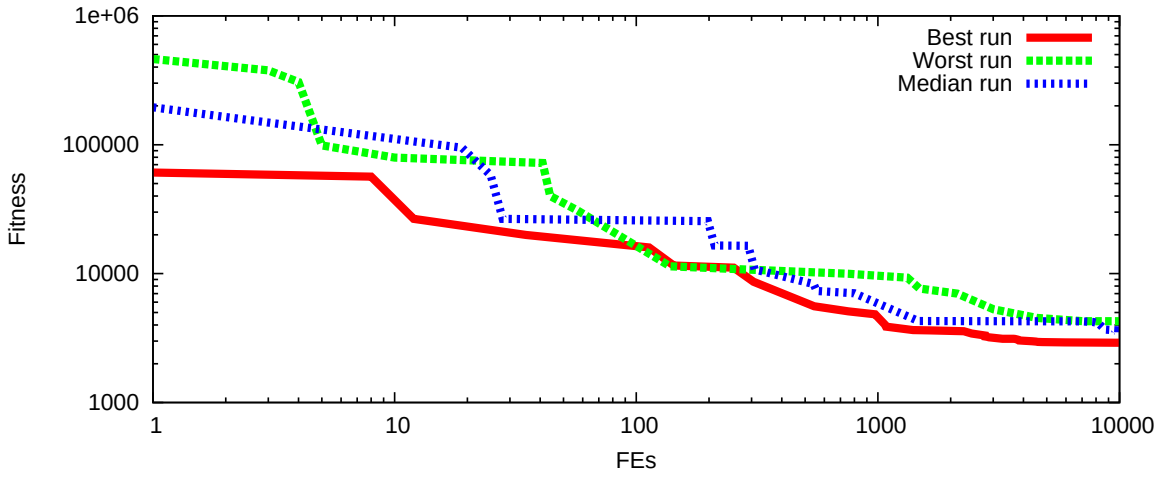
Končne dobljene vrednosti 30 neodvisnih zagonov z različnimi semeni naključnega generatorja, za 10 različnih nastavitev velikosti populacije NP v evolucijskem algoritmu jDE vidimo v tabeli 4.1 in na sliki 4.2. Najboljši povprečni končni približek rešitve dobimo pri $NP = 100$ in ima oceno ovrednoteno s 3031,9, standardna deviacija ocen za zagone pri $NP = 100$ je 184,7. Najslabša ocena zagonov jDE je 4379, pri $NP = 500$. Vzorčene proceduralne modele prvega zagona jDE pri $NP = 100$, s semenom naključnega generatorja 1, vidimo na sliki 4.1. Drevo na sliki je visoko 2,5 m, od tega je višina debla 1 m, zato se upodobitev razteza le na delu slikovnega platna, ki v višino zajema skupno 25 m. Zato so slike z originalno ločjivostjo 250x250 na sliki 4.1 obrezane in povečane, da je prikazan le del slike, ki vsebuje upodobitev drevesa. Iz slike 4.1 vidimo, da je izbrana ločljivost dovolj visoka za rekonstrukcijo proceduralnih drevesnih modelov.

Najboljši dosežen približek gojitvenemu semenu iskanega drevesa po koncu optimizacije ima $S = 43$ žil v deblu enako kot seme drevesa iz referenčne slike. Kot filotakse najdenega semena je $\alpha_{\mathrm{p}} = 260°$, v referenčnem semenu znaša vrednost 262°. Ta parametra sta primer, da je naš postopek zmožen doseči celo delno pokrivanje genotipov rekonstruiranega modela in referenčnega drevesa, čeprav na začetku iskanja genotipe inicializiramo uniformno naključno v vsem definicijskem območju. Iz slike 4.3 vidimo

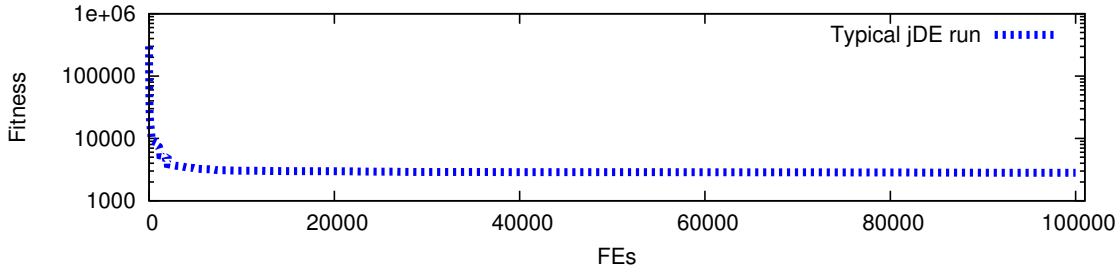
Slika 4.3: Slika najboljšega najdenega približka (levo) in referenčna slika (desno).

tudi, da semeni upodabljata na videz dokaj podobni drevesi, kar dodatno potrdi izbiro uporabljenih metrik. Izidi rekonstrukcije kažejo tudi, da izbrano mejno največje število žil $S = 410$, dimenzija vzorčenja 250x250 in druge nastavitve uporabljenih parametrov lahko z našim pristopom dosežejo obetavne optimizacijske rezultate.

Konvergenca evolucijskih zagonov z najboljšo, najslabšo in medialno doseženo oceno pri koncu optimizacije z algoritmom jDE za vse testirane velikosti populacij NP vidimo na sliki 4.4. V najboljšem zagonu algoritem konvergira dokaj hitro od ocene 107553 vse do 2964 in pri tem doseže večino optimizacij že pred 4000 FEs, kar nakazuje na to, da je zastavljena meja 10000 FEs dovolj za naš optimizacijski pristop. Iz slike 4.5



Slika 4.4: Konvergenca najboljšega, medialnega in najslabšega evolucijskega zagona za algoritem jDE.



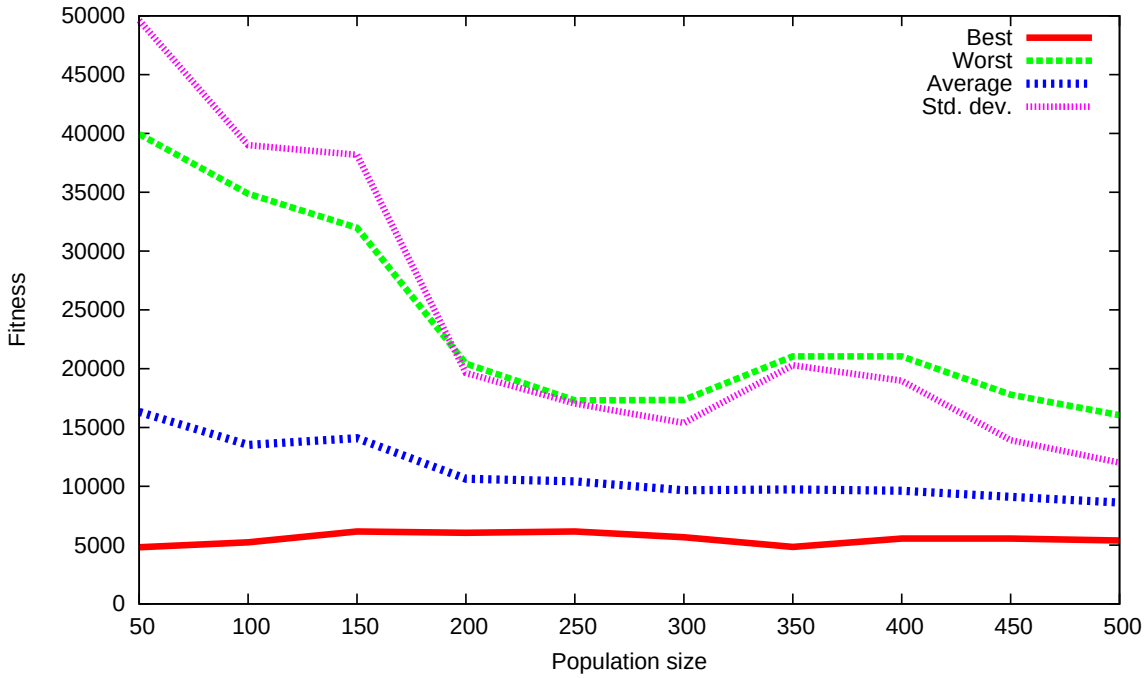
Slika 4.5: Konvergenca tipičnega zagona algoritma jDE z $NP = 100$, podaljšan do 100000 FEs.

Tabela 4.1: Doseženi rezultati za DE in jDE pri različnih velikostih populacije.

NP	Najboljši		Najslabši		Povprečje		Std. dev.	
	DE	jDE	DE	jDE	DE	jDE	DE	jDE
50	**4822**	**2898**	39950	3664	16328,7	3044,1	49626,8	982,2
100	5243	2964	34872	**3098**	13525,2	**3031,9**	39006,6	**184,7**
150	6160	3073	31981	3300	14092,8	3175,7	38177,4	309,7
200	6049	3113	20434	3585	10637,4	3310,4	19630,9	608,8
250	6160	3243	17301	3634	10424,9	3400,0	17039,1	540,1
300	5676	3272	17343	4022	9666,6	3580,6	15389,1	915,9
350	4850	3341	21050	4035	9733,6	3660,2	20290,5	1017,3
400	5554	3405	21050	4207	9617,8	3829,0	18981,2	1047,0
450	5554	3450	17796	4364	9111,2	3874,1	13917,7	1281,7
500	5388	3380	**16051**	4379	**8619,4**	4001,6	**12006**	1475,3

je jasno videti, da algoritem jDE pri $NP = 100$ v podaljšanem poskusu z največjim dovoljenim številom ovrednotenj kriterijske funkcije, 100000 FEs, glede na uporabljeno metriko ne napreduje bistveno po presegu zastavljene osnovne meje FEs, 10000 FEs.

Da bi statistično primerjali zmogljivost našega pristopa z algoritmom jDE napram osnovnemu algoritmu DE s fiksnimi krmilnimi parametri $F = 0{,}5$ in $CR = 0{,}9$ ter strategijo DE/rand/1/bin, smo oba pognali nad istim testom. Rezultati zmogljivosti algoritma DE za naš test so v tabeli 4.1 oz. grafu na sliki 4.6. Kot vidimo, je merjena zmogljivost nad uporabljenim testom za naš pristop z algoritmom jDE vedno boljša od pristopa z osnovnim algoritmom DE. Najboljša dosežena ocena z algoritmom DE je 4822, pri $NP = 50$. Najboljša povprešna ocena z algoritmom DE je 8619,43, dosežena pri $NP = 500$, kjer je standardna deviacija enaka 12006. Iz tabele 4.1 in primerjave



Slika 4.6: Ocene zmogljivosti algoritma DE, odvisne od velikosti populacije.

slik 4.2 in 4.6 vidimo, da so najboljši rezultati algoritma DE vedno slabši od rezultatov algoritma jDE pri vseh navedenih velikostih populacij. Znatna razlika v zmogljivosti med algoritmoma na našem testu zato nakazuje, da je uporaba algoritma jDE s samoprilagodljivimi krmilnimi parametri na našem optimizacijskem problemu ustreznejša od algoritma DE s fiksnimi krmilnimi parametri.

4.3 Razprava o dobljenih rezultatih

Ker smo pokazali, da je s predlaganim proceduralnim modelom po kodiranju v genotip možno izvesti evolucijski proces iskanja parametriziranih proceduralnih modelov, razvidno iz postavljenih definicij v prejšnjem poglavju in slike 4.1 med podanimi rezultati preskusov, potrdimo prvo hipotezo teze te doktorske disertacije.

S postavitvijo ustreznostne funkcije in preslikavo genotipa v fenotip smo pokazali, da je algoritem diferencialne evolucije primeren za iterativno rekonstrukcijo izbranega parametriziranega proceduralnega modela, kot je vidno iz konvergenčnih grafov na sliki 4.4, stabilnosti glede velikost populacije iz slike 4.2 in iterativnega izboljševanja na sliki 4.1. S tem smo potrdili drugo hipotezo. Vendar, pokazali smo še več, namreč, diferencialna evolucija je še posebej primerna za evolucijo izbranega modela, saj je ta fiksne dimenzije, parametre pa smo kodirali v domeno realnih števil, za katero je algoritem DE posebej učinkovito načrtovan.

S primerjavo osnovnega algoritma DE brez samoprilagajanja krmilnih parametrov in algoritma jDE s samoprilagodljivimi krmilnimi parametri smo pokazali, da je algoritem jDE za naš optimizacijski postopek ustreznejši, kar je vidno iz tabele 4.1. S tem potrdimo še zadnjo, tretjo hipotezo teze. Ker smo potrdili vse tri zadane hipoteze, s tem potrdimo tezo te disertacije.

5. Zaključek

Ideas shape the course of history. (John Maynard Keynes)

V tej disertaciji smo zadano tezo o lastnem razvoju novega evolucijskega algoritma, ki omogoča modeliranje z rekonstrukcijo parametriziranih proceduralnih modelov olesenelih rastlin iz slikovnih projekcij referenčnih modelov, utemeljili s potrditvijo treh hipotez. Pokazali smo, da je s predlaganim proceduralnim modelom po kodiranju v genotip možno izvesti evolucijski proces iskanja parametriziranih proceduralnih modelov. S postavitvijo ustreznostne funkcije in preslikavo genotipa v fenotip smo pokazali, da je algoritem diferencialne evolucije primeren za iterativno rekonstrukcijo izbranega parametriziranega proceduralnega modela. Pokazali smo, da je algoritem diferencialne evolucije še posebej primeren za evolucijo izbranega modela, saj je ta fiksne dimenzije, parametre pa smo kodirali v domeno realnih števil, za katero je algoritem diferencialne evolucije posebej učinkovito načrtovan. Pokazali smo še, da je algoritem jDE s samoprilagodljivimi krmilnimi parametri za naš optimizacijski postopek ustreznejši od osnovnega algoritma DE brez samoprilagajanja krmilnih parametrov. Dobljeni rezultati potrjujejo, da je predstavljen pristop primeren za modeliranje drevesnih rastlin za računalniško animacijo, s pomočjo evolucije numerično kodiranega proceduralnega modela.

S tem ugotavljamo, da teza doktorske disertacije pritrdilno utemeljuje zastavljene hipoteze. Izsledke, ki smo jih pokazali v tej doktorski disertaciji, smo objavili tudi širši znanstveni javnosti, predvsem z revijalnim prispevkom [267].

V nadaljevanju raziskovalnega dela na obravnavanem področju bi bilo možno raziskati primerjalne metrike s poudarkom na topoloških podobnostih dreves in jih uporabiti v načrtovanju z večkriterijsko optimizacijo. Matrične parametre v genotipih posameznikov bi lahko skušali kodirati tudi s pomožnimi vektorskimi parametri. Implementacijo pristopa bi bilo dobro razširiti tudi na področje vseprisotnega računalništva in razširjene resničnosti.

A. Statistično ovrednotenje naših algoritmov diferencialne evolucije

Now you wouldn't believe me if I told you, but I could run like the wind blows.
From that day on, if I was going somewhere, I was running! (Forrest Gump)

Ker smo jih pripravili v sklopu priprav na disertacijo, k tej razpravi dodajamo nekatere analize krmilnih parametrov in ocene kakovosti svojih algoritmov ter jih primerjamo z najboljšimi algoritmi trenutno na voljo.

Tabela A.1: Povprečne vrednosti F in CR za primer zagona algoritmov jDE in SA-DE.

	jDE		SA-DE	
Fun.	F	CR	F	CR
f_1	0,22±0,11	0,50±0,25	0,35±0,06	0,54±0,11
f_2	0,21±0,12	0,45±0,24	0,36±0,08	0,51±0,11
f_3	0,31±0,18	0,90±0,19	0,72±0,11	0,48±0,08
f_4	0,22±0,13	0,44±0,25	0,39±0,07	0,18±0,04
f_5	0,30±0,17	0,69±0,32	0,38±0,06	0,67±0,13
f_6	0,21±0,11	0,49±0,27	0,47±0,16	0,36±0,06
f_7	0,24±0,10	0,54±0,28	0,48±0,13	0,36±0,05
f_8	0,32±0,23	0,36±0,30	0,48±0,21	0,30±0,12
f_9	0,24±0,15	0,35±0,29	0,43±0,09	0,23±0,12
f_{10}	0,23±0,12	0,48±0,25	0,45±0,21	0,39±0,09
f_{11}	0,23±0,11	0,47±0,25	0,36±0,07	0,47±0,09
f_{12}	0,22±0,11	0,48±0,25	0,36±0,06	0,48±0,11
f_{13}	0,22±0,11	0,47±0,25	0,36±0,07	0,51±0,11

Ob predstavitvi algoritma SA-DE [31] nas je zanimalo obnašanje mehanizma prilagajanja krmilnih parametrov in morebitne razlike v dobljenih vrednostih glede na

algoritem jDE [23]. Tabela A.1 prikazuje dobljene povprečne vrednosti krmilnih parametrov F in CR. Primerjane so eksperimentalno dobljene povprečne vrednosti krmilnih parametrov algoritma jDE in algoritma SA-DE. Kot vidimo, se vrednosti krmilnih parametrov med algoritmama razlikujejo in so za algoritem SA-DE drugače karakteristično vezane z določeno kriterijsko funkcijo.

Ocenili smo tudi kakovost algoritma DEwSAcc [263], tokrat za funkcije z velikimi dimenzijami [220], dimenzij $D = \{100, 500, 1000\}$. Rezultati optimizacije so vidni v tabeli A.2. Iz rezultatov tekmovanja [219] razberemo, da je algoritem DEwSAcc po kakovosti primerljiv z naborom tekmovalnih algoritmov.

V tabeli A.3 podajamo povzetek v [255] pripravljenih rezultatov primerjav ocen algoritma DEMOwSA [260] in drugih iz tekmovalne sekcije CEC 2007, pri 5e+5 ovrednotenjih vsake od kriterijskih funkcij [98]. Tabela za vsak z DEMOwSA primerjan algoritem prikazuje število boljših, slabših in nesignifikantnih razlik pri indikatorjih I_R in $I_{\overline{H}}$. V zadnjem stolpcu je navedena še vsota obeh razlik v vrednostih indikatorjev. Vnosi v tabeli so urejeni po tej vsoti, ki je tudi naš skupni metakriterij in njegove pozitivne vrednosti so boljše za algoritem DEMOwSA. Kot vidimo, je algoritem DEMOwSA v povprečju boljši od prvih šestih algoritmov, zadnja dva v tabeli pa v povprečju na danem testnem ogrodju dajeta nekoliko boljše rezultate. Omenimo še, da smo ocene indikatorjev za primerjane algoritme dobili iz člankov, ki so preizkušali primerjane algoritme. Le za osnovni algoritem DEMO smo uporabili enako inicializirano populacijo vektorjev ter fiksne krmilne parametre in izvedli potrebne simulacije, saj zanj potrebni eksperimentalni rezultati pred našo primerjavo še niso bili znani.

Konvergenčne vrednosti krmilnih parametrov algoritma DEMOwSA smo prav tako analizirali, kot tudi njihovo dinamiko [255]. Tabela A.4 podaja povprečja konvergenčnih vrednosti krmilnih parametrov F in CR pri vsaki od funkcij do 5e+5 ovrednotenj. Posamezna vrednost predstavlja povprečje (oz. v oklepajih standardni odklon) krmilnega parametra skozi zadnjih 25 generacij za vse neodvisne zagone, za primerke, ki so izboljšali starševsko rešitev. Kot vidimo, se krmilna parametra samoprilagodita na precej različne vrednosti, zato ni možno sklepati o fiksni vrednosti posameznega parametra, ki bi enako dobro delovala za vse funkcije. Pri tej meritvi smo namreč uporabili začetno vrednost $CR_{\text{init}} = 0{,}3$ in pri vseh funkcijah vidimo, da te ni med njimi, kar dodatno nakazuje na uspešno samoprilagajanje krmilnih parametrov.

Ker je za nekatere izmed funkcij dinamika krmilnih parametrov F in CR precej različna, smo v [255, 264] analizirali še dinamiko krmilnih parametrov in ugotovili, da

Tabela A.2: Za [220] na $D = \{100, 500, 1000\}$ dosežene vrednosti algoritma DEwSAcc.

FES		1	2	3	4	5	6	7
$D = 100$								
5, 00e + 3	1. (Najboljša)	8,3828e+04	1,0821e+02	1,7758e+10	1,1226e+03	8,0886e+02	1,9204e+01	-9,3759e+02
	7.	1,0587e+05	1,1488e+02	2,5129e+10	1,2271e+03	8,9186e+02	1,9726e+01	-8,9853e+02
	13. (Mediana)	1,2259e+05	1,1837e+02	2,9213e+10	1,2663e+03	9,9249e+02	2,0121e+01	-8,9062e+02
	19.	1,3247e+05	1,2374e+02	3,7742e+10	1,2843e+03	1,0451e+03	2,0278e+01	-8,8153e+02
	25. (Najslabša)	1,5740e+05	1,3191e+02	5,7587e+10	1,3393e+03	1,4357e+03	2,0582e+01	-8,6589e+02
	Povprečje	1,2067e+05	1,1918e+02	3,2692e+10	1,2504e+03	9,9834e+02	2,0015e+01	-8,9184e+02
	Std	2,1065e+04	5,8203	1,0456e+10	5,2740e+01	1,4831e+02	3,8639e-01	1,5851e+01
5, 00e + 4	1. (Najboljša)	1,1883	4,8652e+01	6,2478e+03	2,3028e+02	3,1522e-01	1,9315e-01	-1,1567e+03
	7.	9,0535	5,2743e+01	2,9974e+04	2,5919e+02	1,0790	1,3143	-1,1338e+03
	13. (Mediana)	1,7788e+01	5,4268e+01	4,9348e+04	2,9609e+02	1,1649	1,8422	-1,1234e+03
	19.	2,6560e+01	5,7485e+01	9,8943e+04	3,1490e+02	1,2311	2,2366	-1,1053e+03
	25. (Najslabša)	1,3695e+02	6,8450e+01	4,6189e+05	4,0082e+02	1,8460	3,2108	-1,0957e+03
	Povprečje	2,4078e+01	5,5515e+01	8,3301e+04	2,9450e+02	1,1476	1,6941	-1,1234e+03
	Std	2,7276e+01	4,3898	9,3131e+04	4,4907e+01	2,9912e-01	8,0548e-01	1,8137e+01
5, 00e + 5	1. (Najboljša)	5,6843e-14	3,1796	9,0367e+01	5,6843e-14	2,8421e-14	8,5265e-14	-1,4137e+03
	7.	5,6843e-14	4,6757	9,3053e+01	3,7096e-10	2,8421e-14	1,1368e-13	-1,3814e+03
	13. (Mediana)	5,6843e-14	6,3670	1,4611e+02	1,9899	2,8421e-14	1,1368e-13	-1,3664e+03
	19.	5,6843e-14	1,0622e+01	1,5296e+02	4,9747	2,8421e-14	1,1368e-13	-1,3460e+03
	25. (Najslabša)	5,6843e-14	2,7417e+01	3,0627e+02	3,5818e+01	5,6843e-14	1,4210e-13	-1,3204e+03
	Povprečje	5,6843e-14	8,2500	1,4463e+02	4,3778	3,0695e-14	1,1255e-13	-1,3650e+03
	Std	0,0000	5,3274	5,8483e+01	7,6532	7,8696e-15	1,5306e-14	2,4565e+01
$D = 500$								
2, 50e + 4	1. (Najboljša)	1,2787e+06	1,4415e+02	5,3248e+11	7,9014e+03	1,1023e+04	2,0797e+01	-3,6612e+03
	7.	1,3741e+06	1,4830e+02	6,9256e+11	8,3009e+03	1,1375e+04	2,0984e+01	-3,6249e+03
	13. (Mediana)	1,4301e+06	1,5140e+02	7,3432e+11	8,4198e+03	1,1580e+04	2,1033e+01	-3,5965e+03
	19.	1,4713e+06	1,5539e+02	8,0235e+11	8,5105e+03	1,2141e+04	2,1062e+01	-3,5751e+03
	25. (Najslabša)	1,5263e+06	1,5912e+02	9,9182e+11	8,6877e+03	1,3536e+04	2,1130e+01	-3,5581e+03
	Povprečje	1,4215e+06	1,5157e+02	7,5376e+11	8,3908e+03	1,1749e+04	2,1019e+01	-3,6030e+03
	Std	6,5764e+04	4,3153	9,2525e+10	1,8565e+02	5,7805e+02	7,5075e-02	3,1426e+01
2, 50e + 5	1. (Najboljša)	5,3448e+04	1,0648e+02	7,6260e+09	3,9041e+03	4,2257e+02	1,1718e+01	-4,4236e+03
	7.	5,8422e+04	1,1114e+02	9,4815e+09	4,0264e+03	4,8057e+02	1,2121e+01	-4,3635e+03
	13. (Mediana)	6,2486e+04	1,1277e+02	1,0598e+10	4,1424e+03	5,1518e+02	1,2252e+01	-4,3209e+03
	19.	6,6214e+04	1,1733e+02	1,1713e+10	4,2123e+03	5,7750e+02	1,2751e+01	-4,2938e+03
	25. (Najslabša)	1,0434e+05	1,2184e+02	1,4378e+10	4,3296e+03	7,9815e+02	1,4105e+01	-4,2581e+03
	Povprečje	6,5378e+04	1,1375e+02	1,0733e+10	4,1260e+03	5,4453e+02	1,2562e+01	-4,3293e+03
	Std	1,2098e+04	3,9597	1,6030e+09	1,2645e+02	9,2209e+01	6,8792e-01	4,7115e+01
2, 50e + 6	1. (Najboljša)	5,5194e-11	6,9437e+01	1,3387e+03	2,8699e+02	1,1482e-11	7,0973e-07	-6,5747e+03
	7.	2,5283e-10	7,3939e+01	1,6610e+03	3,2619e+02	3,3054e-11	4,7260e-06	-5,7378e+03
	13. (Mediana)	8,5242e-10	7,5286e+01	1,7515e+03	3,4896e+02	8,5577e-11	1,2709e-05	-5,7129e+03
	19.	1,1424e-09	7,7271e+01	1,9962e+03	4,0069e+02	2,5801e-10	9,2623e-01	-5,6951e+03
	25. (Najslabša)	1,8813e-08	8,2208e+01	2,5213e+03	4,6943e+02	9,8572e-03	1,5436	-5,6009e+03
	Povprečje	2,0958e-09	7,5737e+01	1,8130e+03	3,6403e+02	6,9013e-04	4,8041e-01	-5,7487e+03
	Std	4,6182e-09	3,0465	2,7425e+02	5,2353e+01	2,4149e-03	5,7362e-01	1,8370e+02
$D = 1000$								
5, 00e + 4	1. (Najboljša)	3,1517e+06	1,5282e+02	1,6805e+12	1,7615e+04	2,8321e+04	2,1129e+01	-6,8913e+03
	7.	3,2855e+06	1,5926e+02	1,7836e+12	1,7851e+04	2,9085e+04	2,1181e+01	-6,8048e+03
	13. (Mediana)	3,3457e+06	1,6153e+02	1,9141e+12	1,8020e+04	2,9650e+04	2,1197e+01	-6,7939e+03
	19.	3,3836e+06	1,6326e+02	2,0498e+12	1,8244e+04	3,0190e+04	2,1213e+01	-6,7617e+03
	25. (Najslabša)	3,5443e+06	1,6593e+02	2,2811e+12	1,8635e+04	3,1439e+04	2,1246e+01	-6,7069e+03
	Povprečje	3,3445e+06	1,6104e+02	1,9115e+12	1,8055e+04	2,9707e+04	2,1199e+01	-6,7859e+03
	Std	8,7727e+04	3,2054	1,5648e+11	2,4394e+02	8,3760e+02	2,8576e-02	3,9072e+01
5, 00e + 5	1. (Najboljša)	4,9622e+05	1,2195e+02	1,3986e+11	1,0738e+04	4,2273e+03	1,7338e+01	-7,9377e+03
	7.	5,1087e+05	1,2549e+02	1,4776e+11	1,0927e+04	4,5186e+03	1,7655e+01	-7,8972e+03
	13. (Mediana)	5,2243e+05	1,2879e+02	1,5336e+11	1,0991e+04	4,7242e+03	1,7759e+01	-7,8451e+03
	19.	5,5952e+05	1,3082e+02	1,6018e+11	1,1109e+04	4,8783e+03	1,7948e+01	-7,8152e+03
	25. (Najslabša)	6,2241e+05	1,3645e+02	1,8745e+11	1,1471e+04	5,6358e+03	1,8461e+01	-7,7558e+03
	Povprečje	5,3604e+05	1,2825e+02	1,5510e+11	1,1025e+04	4,7567e+03	1,7799e+01	-7,8549e+03
	Std	3,5192e+04	3,4718	1,1013e+10	1,9433e+02	3,3061e+02	2,6880e-01	5,0918e+01
5, 00e + 6	1. (Najboljša)	3,1715e-03	9,2531e+01	6,9294e+03	1,5824e+03	2,3156e-04	1,5605	-1,1508e+04
	7.	5,0898e-03	9,4671e+01	8,2666e+03	1,7299e+03	4,3463e-04	2,1076	-1,0520e+04
	13. (Mediana)	8,0634e-03	9,6356e+01	9,0116e+03	1,8170e+03	6,7930e-04	2,3699	-1,0414e+04
	19.	9,7046e-03	9,7274e+01	1,0035e+04	1,8985e+03	1,5052e-03	2,5210	-1,0358e+04
	25. (Najslabša)	2,6490e-02	9,9238e+01	1,1629e+04	2,0998e+03	2,0272e-02	2,7450	-1,0268e+04
	Povprečje	8,7874e-03	9,6058e+01	9,1498e+03	1,8239e+03	3,5826e-03	2,2956	-1,0585e+04
	Std	5,2705e-03	1,8194	1,2605e+03	1,3773e+02	5,7398e-03	2,9834e-01	4,1846e+02

Tabela A.3: Povzetek rezultatov z DEMOwSA primerjanih algoritmov pri 5e+5 ovrednotenjih kriterijske funkcije.

Ime	Vir	Indikator I_R			Indikator $I_{\overline{H}}$			Vsota
		Boljše	Slabše	Nesig.	Boljše	Slabše	Nesig.	
GDE3	[126]	6	12	1	6	13	0	*-13*
NSGAII-SBX	[208]	8	11	0	9	10	0	*-4*
NSGAII-PBX	[127]	10	9	0	10	8	1	**3**
MOSaDE	[100]	9	10	0	12	7	0	**4**
DEMO	[201]	11	7	1	10	5	4	**9**
MO_PSO	[280]	11	6	2	10	6	3	**9**
MTS	[224]	11	7	1	12	7	0	**9**
MO_DE	[283]	14	5	0	13	6	0	**16**

Tabela A.4: Vrednosti krmilnih parametrov po 5e+5 ovrednotenjih kriterijske funkcije.

Zap. št.	Funkcija (vir: [98])	$\overline{F}_{\text{imp}}$	$\overline{CR}_{\text{imp}}$
1.	OKA2	0,53 (0,32)	0,40 (0,19)
2.	SYMPART	0,17 (0,06)	0,75 (0,16)
3.	S_ZDT1	0,26 (0,08)	0,55 (0,14)
4.	S_ZDT2	0,31 (0,09)	0,56 (0,14)
5.	S_ZDT4	0,32 (0,10)	0,55 (0,14)
6.	R_ZDT4	0,45 (0,23)	0,26 (0,17)
7.	S_ZDT6	0,55 (0,00)	0,40 (0,00)
8.	S_DTLZ2_M3	0,32 (0,09)	0,49 (0,13)
9.	R_DTLZ2_M3	0,68 (0,15)	0,35 (0,11)
10.	S_DTLZ3_M3	0,33 (0,10)	0,48 (0,12)
11.	WFG1_M3	0,52 (0,16)	0,24 (0,08)
12.	WFG8_M3	0,35 (0,12)	0,46 (0,13)
13.	WFG9_M3	0,63 (0,18)	0,31 (0,10)
14.	S_DTLZ2_M5	0,36 (0,11)	0,32 (0,09)
15.	R_DTLZ2_M5	0,61 (0,16)	0,23 (0,07)
16.	S_DTLZ3_M5	0,62 (0,16)	0,06 (0,02)
17.	WFG1_M5	0,63 (0,16)	0,24 (0,07)
18.	WFG8_M5	0,54 (0,15)	0,40 (0,10)
19.	WFG9_M5	0,66 (0,16)	0,27 (0,08)

Tabela A.5: Povzetek ocen rezultatov algoritmov.

Ime algoritma	Vir	Indikator $I_{\overline{H}}$		
		Boljše	Slabše	Nesignif.
GDE3	[126]	**9**	10	0
NSGAII-SBX	[208]	**10**	9	0
MO_DE	[280]	11	7	1
NSGAII-PBX	[127]	**12**	7	0
MOSaDE	[100]	**13**	6	0
MTS	[224]	**13**	6	0
MO_PSO	[283]	**13**	6	0

vrednosti pogosto menjajo smer samoprilagajanja, saj algoritem menjuje faze lokalnega in globalnega iskanja, ki ju krmilna parametra nadzorujeta. Iz analize dinamike krmilnih parametrov F in CR je razvidno, da se ti precej spreminjajo in jih ni moč nastaviti enako za poljuben problem, samoprilagajanje še toliko bolj pomaga pri reševanju industrijskih problemov [44]. Za te probleme praviloma pogosto velja, da ročno nastavljanje krmilnih parametrov in znovični zagon eksperimentov z ovrednotenji posameznih poskusnih rešitev z novimi parametri lahko podraži razvoj izdelka zaradi vključenega potrošnega materiala in podaljšanega razvojnega časa. Z uporabo samoprilagajanja smo povečali robustnost algoritma na večini testnih primerov, t.j. da je algoritem z večjo verjetnostjo uspešno reševal optimizacijske probleme. To statistično potrjuje signifikantni vpliv, kot tudi povečanje učinkovitosti optimizacije zaradi vpeljanega mehanizma samoprilagajanja krmilnih parametrov.

Optimizacijsko učinkovitost za enega od indikatorjev na izbranih problemih smo glede na algoritem DEMOwSA še dodatno izboljšali z algoritmom DEMOwSA-SQP. Rezultate nad istim preizkusnim naborom vidimo v tabeli A.5, kjer so izboljšane vrednosti indikatorja označene z odebeljenim tiskom. Kot vidimo iz tabele, je predstavljen algoritem tako rangiran na drugem mestu.

Algoritem DECMOSA-SQP [265] smo ocenili na testnem ogrodju [278] in daje rezultate, kot vidimo v tabeli A.6. Komentarje dobljenih vrednosti glede na druge algoritme, razvrstitev algoritmov in skupno oceno najdemo v poročilu [277]. V tem poročilu je algoritem DECMOSA-SQP razvrščen na različna mesta na metakriterijskih lestvicah in prav tako primerljiv s številnimi aktualnimi algoritmi.

Tabela A.6: Povprečne vrednosti metrike IGD za 30 končnih aproksimacijskih množic dobljenih za vsak problem iz testnega nabora [278].

Funkcija	Vrednost IGD
UF1	0,0770281
UF2	0,0283427
UF3	0,0935006
UF4	0,0339266
UF5	0,167139
UF6	0,126042
UF7	0,024163
UF8	0,215834
UF9	0,14111
UF10	0,369857
CF1	0,107736
CF2	0,0946079
CF3	1000000
CF4	0,152659
CF5	0,412756
CF6	0,147824
CF7	0,26049
CF8	0,176344
CF9	0,127132
CF10	0,507051
R2_DTLZ2_M5	0,383041
R2_DTLZ3_M5	943,352
WFG1_M5	1,9178

Problematiko optimizacije realnih industrijskih problemov [53] smo naslovili z algoritmo $\text{jDE}_{\text{NP,MM}}$ [258]. Primerjavo povprečnih najboljših dobljenih vrednosti optimizacije za 25 neodvisnih zagonov algoritma $\text{jDE}_{\text{NP,MM}}$ in drugih algoritmov iz [53] vidimo v tabeli A.7 za funkcije 1-11 in v tabeli A.8 še za funkcije 12-22 in skupno oceno. V tretji koloni so povprečne vrednosti končnih rezultatov našega algoritma, v četrti algoritma avtorjev Korošec in Šilc [120, 24] z oznako C-0036 [121] in peti algoritma avtorjev Mallipeddi in Suganthan z oznako C-0362 [146]. V šesti in sedmi koloni sta navedeni razliki

med našim $\text{jDE}_{\text{NP,MM}}$ in vsakim od primerjanih algoritmomov. Iz primerjave razberemo, da je algoritem $\text{jDE}_{\text{NP,MM}}$ 47 krat boljši od algoritma z oznako C-0036 in 31 krat boljši od algoritma z oznako C-0362. Na podlagi rezultatov primerjave lahko uvrstimo algoritem $\text{jDE}_{\text{NP,MM}}$ ob bok najboljšim samoprilagodljivim evolucijskim algoritmov iz tekmovanja [53].[1]

[1] O vplivnosti naših predstavljenih izboljšav algoritmov diferencialne evolucije pričajo tudi citati naših objav, skupno število citatov v bazi SCOPUS je trenutno (15. maj 2012) enako 80.

Tabela A.7: Primerjava algoritma $jDE_{NP,MM}$ za funkcije 1-11 iz [53].

Fun.	FES	$jDE_{NP,MM}$	C-0036 [121]	C-0362 [146]	diff(C-0036)	diff(C-0362)
F1	50000	7,9306e+00	1,3995e+01	7,06E+00	**-6,0644e+00**	*8,7060e-01*
F1	100000	1,5262e+00	1,0872e+01	2,29E+00	**-9,3458e+00**	**-7,6380e-01**
F1	150000	1,1067e+00	1,0128e+01	1,78E+00	**-9,0213e+00**	**-6,7330e-01**
F2	50000	-2,0288e+01	-1,5775e+01	-1,26E+01	**-4,5130e+00**	**-7,6880e+00**
F2	100000	-2,2839e+01	-1,6766e+01	-1,64E+01	**-6,0730e+00**	**-6,4390e+00**
F2	150000	-2,4284e+01	-1,7566e+01	-1,83E+01	**-6,7180e+00**	**-5,9840e+00**
F3	50000	1,1515e-05	1,1515e-05	1,15E-05	0,0000e+00	*1,5000e-08*
F3	100000	1,1515e-05	1,1515e-05	1,15E-05	0,0000e+00	*1,5000e-08*
F3	150000	1,1515e-05	1,1515e-05	1,15E-05	0,0000e+00	*1,5000e-08*
F4	50000	1,5316e+01	1,4173e+01	1,67E+01	*1,1430e+00*	**-1,3840e+00**
F4	100000	1,5158e+01	1,4039e+01	1,67E+01	*1,1190e+00*	**-1,5420e+00**
F4	150000	1,5158e+01	1,3936e+01	1,67E+01	*1,2220e+00*	**-1,5420e+00**
F5	50000	-3,3386e+01	-3,3533e+01	-2,38E+01	*1,4700e-01*	**-9,5860e+00**
F5	100000	-3,4374e+01	-3,3834e+01	-2,75E+01	**-5,4000e-01**	**-6,8740e+00**
F5	150000	-3,4779e+01	-3,3909e+01	-2,90E+01	**-8,7000e-01**	**-5,7790e+00**
F6	50000	-2,7050e+01	-2,3150e+01	-1,28E+01	**-3,9000e+00**	**-1,4250e+01**
F6	100000	-2,8021e+01	-2,5581e+01	-1,55E+01	**-2,4400e+00**	**-1,2521e+01**
F6	150000	-2,8651e+01	-2,6748e+01	-1,70E+01	**-1,9030e+00**	**-1,1651e+01**
F7	50000	1,3231e+00	1,0262e+00	1,61E+00	*2,9690e-01*	**-2,8690e-01**
F7	100000	1,1477e+00	9,6956e-01	1,49E+00	*1,7814e-01*	**-3,4230e-01**
F7	150000	1,1677e+00	9,3895e-01	1,42E+00	*2,2875e-01*	**-2,5230e-01**
F8	50000	2,2000e+02	2,2000e+02	2,20E+02	0,0000e+00	0,0000e+00
F8	100000	2,2000e+02	2,2000e+02	2,20E+02	0,0000e+00	0,0000e+00
F8	150000	2,2000e+02	2,2000e+02	2,20E+02	0,0000e+00	0,0000e+00
F9	50000	3,2413e+03	1,6940e+03	2,875E+03	*1,5473e+03*	*3,6630e+02*
F9	100000	2,5200e+03	1,2338e+03	2,529E+03	*1,2862e+03*	**-9,0000e+00**
F9	150000	2,2419e+03	1,0692e+03	2,529E+03	*1,1727e+03*	**-2,8710e+02**
F10	50000	-1,9870e+01	-1,2655e+01	-1,52E+01	**-7,2150e+00**	**-4,6700e+00**
F10	100000	-2,0743e+01	-1,3213e+01	-1,55E+01	**-7,5300e+00**	**-5,2430e+00**
F10	150000	-2,1300e+01	-1,3540e+01	-1,56E+01	**-7,7600e+00**	**-5,7000e+00**
F11	50000	2,2755e+05	5,2607e+04	5,26E+04	*1,7494e+05*	*1,7495e+05*
F11	100000	8,3780e+04	5,2160e+04	5,24E+04	*3,1620e+04*	*3,1380e+04*
F11	150000	5,3040e+04	5,2017e+04	5,22E+04	*1,0230e+03*	*8,4000e+02*

Tabela A.8: Primerjava algoritma $jDE_{NP,MM}$ za funkcije 12-22 iz [53] in skupna ocena.

Fun.	FES	$jDE_{NP,MM}$	C-0036 [121]	C-0362 [146]	diff(C-0036)	diff(C-0362)
F12	50000	1,1424e+06	1,2750e+06	1,08E+06	**-1,3260e+05**	*6,2400e+04*
F12	100000	1,0803e+06	1,2733e+06	1,07E+06	**-1,9300e+05**	*1,0300e+04*
F12	150000	1,0745e+06	1,2717e+06	1,07E+06	**-1,9720e+05**	*4,5000e+03*
F13	50000	1,5452e+04	1,5516e+04	1,55E+04	**-6,4000e+01**	**-4,8000e+01**
F13	100000	1,5446e+04	1,5512e+04	1,55E+04	**-6,6000e+01**	**-5,4000e+01**
F13	150000	1,5444e+04	1,5511e+04	1,55E+04	**-6,7000e+01**	**-5,6000e+01**
F14	50000	1,8666e+04	1,9341e+04	1,82E+04	**-6,7500e+02**	*4,6600e+02*
F14	100000	1,8560e+04	1,9332e+04	1,82E+04	**-7,7200e+02**	*3,6000e+02*
F14	150000	1,8529e+04	1,9323e+04	1,81E+04	**-7,9400e+02**	*4,2900e+02*
F15	50000	3,2882e+04	3,3185e+04	3,28E+04	**-3,0300e+02**	*8,2000e+01*
F15	100000	3,2843e+04	3,3183e+04	3,28E+04	**-3,4000e+02**	*4,3000e+01*
F15	150000	3,2815e+04	3,3181e+04	3,27E+04	**-3,6600e+02**	*1,1500e+02*
F16	50000	1,3652e+05	1,4715e+05	1,32E+05	**-1,0630e+04**	*4,5200e+03*
F16	100000	1,3359e+05	1,4669e+05	1,31E+05	**-1,3100e+04**	*2,5900e+03*
F16	150000	1,3266e+05	1,4666e+05	1,31E+05	**-1,4000e+04**	*1,6600e+03*
F17	50000	1,9650e+06	2,4168e+06	1,92E+06	**-4,5180e+05**	*4,5000e+04*
F17	100000	1,9629e+06	2,1476e+06	1,92E+06	**-1,8470e+05**	*4,2900e+04*
F17	150000	1,9462e+06	2,0375e+06	1,92E+06	**-9,1300e+04**	*2,6200e+04*
F18	50000	9,5964e+05	1,0127e+06	9,44E+05	**-5,3060e+04**	*1,5640e+04*
F18	100000	9,4397e+05	9,4803e+05	9,43E+05	**-4,0600e+03**	*9,7000e+02*
F18	150000	9,4397e+05	9,4569e+05	9,43E+05	**-1,7200e+03**	*9,7000e+02*
F19	50000	1,3137e+06	1,5823e+06	9,94E+05	**-2,6860e+05**	*3,1970e+05*
F19	100000	1,2818e+06	1,4433e+06	9,91E+05	**-1,6150e+05**	*2,9080e+05*
F19	150000	1,2464e+06	1,4012e+06	9,90E+05	**-1,5480e+05**	*2,5640e+05*
F20	50000	9,5964e+05	1,0567e+06	9,44E+05	**-9,7060e+04**	*1,5640e+04*
F20	100000	9,4397e+05	9,8217e+05	9,43E+05	**-3,8200e+04**	*9,7000e+02*
F20	150000	9,4397e+05	9,4887e+05	9,43E+05	**-4,9000e+03**	*9,7000e+02*
F21	50000	1,7961e+01	2,8815e+01	2,26E+01	**-1,0854e+01**	**-4,6390e+00**
F21	100000	1,6921e+01	2,7518e+01	1,98E+01	**-1,0597e+01**	**-2,8790e+00**
F21	150000	1,6599e+01	2,6419e+01	1,88E+01	**-9,8200e+00**	**-2,2010e+00**
F22	50000	1,4754e+01	3,3463e+01	1,99E+01	**-1,8709e+01**	**-5,1460e+00**
F22	100000	1,3154e+01	3,0902e+01	1,57E+01	**-1,7748e+01**	**-2,5460e+00**
F22	150000	1,2404e+01	2,9620e+01	1,39E+01	**-1,7216e+01**	**-1,4960e+00**
B/W					47 / 13	31 / 32

B. Večkriterijska rekonstrukcija

One having most choices is the most flexible one and therefore has most control and power. (NLP)

Kriterijsko metodo naše rekonstrukcije smo razširili tudi na več kriterijev [259]. Pri tem smo zamenjali algoritem jDE avtorjev Brest s sod. [23] z našim že predstavljenim večkriterijskim algoritmom MOjDE [261], ki jDE združuje z algoritmom DEMO [201].

Za primerjave slik smo tokrat izbrali dva kriterija. Metrika za prvi kriterij je vsota manhattanskih razdalj od vsakega črnega (t.j. materialnega) piksla slike do najbližjega črnega piksla referenčne slike in obratno [12]. Metrika za drugi kriterij je število različnih pikslov med slikama. Posamezni fenotip tako ovrednotimo z dvema kriterijema, f_1 in f_2:

$$f_1(\mathbf{x}) = f(\mathbf{g}(\mathbf{x}, \beta_1), \mathbf{g}(\mathbf{x}, \beta_2)) = h_1(\mathbf{z}_1) + h_1(\mathbf{z}_2) \tag{B.1}$$

$$h_1(\mathbf{z}_i) = \sum_{x,y} m_1(z^i_{x,y}, z^*_{i;x,y}) + \sum_{x,y} m_1(z^*_{i;x,y}, z^i_{x,y}) \tag{B.2}$$

$$f_2(\mathbf{x}) = f(\mathbf{g}(\mathbf{x}, \beta_1), \mathbf{g}(\mathbf{x}, \beta_2)) = h_2(\mathbf{z}_1) + h_2(\mathbf{z}_2) \tag{B.3}$$

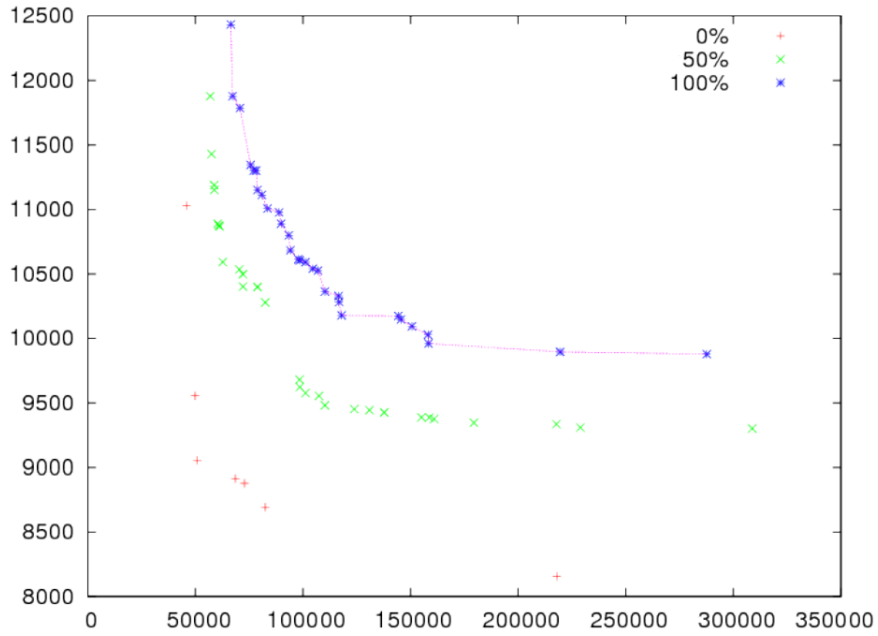
$$h_2(\mathbf{z}_i) = \sum_{x,y} w(z^i_{x,y}, z^*_{i;x,y}) + \sum_{x,y} w(z^*_{i;x,y}, z^i_{x,y}), \tag{B.4}$$

kjer z m_1 označimo funkcijo za izračun manhattanske razdalje do najbližjega piksla v sliki $\mathbf{z}^*$ z vrednostjo 1 (t.j. črno, material) in w, funkcijo za razlikovanje istoležnih pikslov, $i \in \{1, 2\}$.

Večkriterijsko rekonstrukcijo smo preizkusili na isti referenčni sliki za enokriterijsko rekonstrukcijo, vidni iz slike 4.1. Tudi druge parametre smo pustili kot je privzeto nastavljeno v zbranih algoritmih. V tabeli B.1 vidimo kriterijske vektorje v končni aproksimacijski množici nedominiranih kriterijskih vektorjev, ki jih dobimo v vsaj enem od zagonov. Dobljene ploskve dosega (0%, 50% in 100%) tega večkriterijskega evolucijskega procesa za 30 zagonov vidimo na sliki B.1. Na sliki B.2 vidimo slike posameznih izboljšav v evolucijskem procesu za prvi zagon. Na sliki B.3 vidimo končno aproksimacijsko množico velikosti 10 za prvi zagon.

Tabela B.1: Kriterijske vrednosti končne aproksimacijske množice za sedem vektorjev.

Kriterij	1	2	3	4	5	6	7
f_1	45748	49844	50905	68627	72763	82464	217865
f_2	11028	9554	9055	8914	8877	8694	8155



Slika B.1: Ploskve 0%, 50% in 100% dosega za NP=100.

Slika B.2: Upodobitev v trenutni populaciji Pareto optimalnih parametriziranih proceduralnih modelov skozi evolucijo, za prvi zagon.

Slika B.3: Upodobitev končne aproksimacijske množice parametriziranih proceduralnih modelov za prvi zagon.

C. Animacija ekosistema z evolucijskim modelom

It means know thy self. I wanna tell you a little secret, being the one is just like being in love. No one needs to tell you you are in love, you just know it, through and through. (Matrix)

V tem dodatku predstavljamo ogrodje za animacijo razvoja gozdnega ekosistema, ki vsebuje evolucijski model za simulacijo ekosistema. Po zbrani bazi parametričnih modelov za drevesa je namreč postalo jasno, da bi bilo mogoče z njimi vizualizirati tudi drevesne ekosisteme. V ta namen smo pripravili vizualizator terena, ki ima možnost povezave z vizualizatorjem za drevesa in na osnovi simulacije določi porazdelitve dreves po terenu ter na mestih rasti upodobi proceduralni model drevesa. Ker smo vizualizirali rast dreves, je imelo vsako upodobljeno drevo v odvisnosti od svoje starosti na novo določeno geometrijo, z drevesi iste vrste pa si je delilo le parametre proceduralnega modela. V tem dodatku nas tako zanima animacija ekoloških sistemov in za to potrebni modeli. Pri simulatorjih drevesnih ekosistemov se omejimo predvsem na simulatorje, povezane z možnostjo upodabljanja v računalniški grafiki. Ker je ročno nameščanje rastlin na teren zamudno, naključno pa dokaj neprepričljivo, za namestitev dreves v sceni uporabimo simulacijo. V simulaciji ekološkega sistema ustvarimo abstrakten model, ki nam skozi simulacijo daje določene odgovore na različne scenarije. Pri tem je ključnega pomena hitrost in cena odvijanja simulacije. Simulacija nam odgovori v precej krajšem času, kot bi na rezultate čakali v naravnem okolju, ter po nižji ceni in z manj stranskimi učinki.

Osredotočimo se na simulacijo zaraščanja pokrajine. V ta namen potrebujemo ustrezen postopek za porazdelitev dreves po pokrajini, ki upošteva naravne zakonitosti, znane iz ekologije in biologije. S pretvorbo teh zakonitosti v računalniške algoritme omogočimo simulacijo ekoloških zakonitosti v računalniku. Med ekološke vplive v našem abstraktnem pogledu na ekosistem upoštevamo strmino, vlago, vetrovnost in osončenost na mestu rasti rastline kot tudi interakcije med samimi rastlinami. Evolucijski model smo objavili v [269] in zanj prejeli nagrado na tekmovanju IEEE R8SPC.

C.1 Obstoječi evolucijski modeli za animacijo ekosistemov

Z vizualizacijo pokrajin, na katerih so bili prikazani gozdovi in grmovje, se je prvi ukvarjal Holton [96]. Drevesa je predstavil z geometrijskim modelom in jih upodobil z algoritmom sledenja žarku. Ker je njegov model za predstavitev dreves omogočal poenostavitev geometrije, je lahko v pokrajini prikazal več tisoč dreves, ki so bila nameščena ročno.

S 3D teksturami so drevesa predstavili Chiba, Muraoka in Hosokawa [41]. Iz prostorske teksture so z algoritmom sledenja žarku določili gostoto in barvo za voksle. Drevesa so na pokrajino porazdelili s **krožnim Poissonovim vzorčenjem** (*Poisson disc sampling*) [46]. Vsako drevo so tako skušali postaviti na naključno mesto v sceni, preverili pa so le, ali to morda ni preblizu kakšnemu že postavljenemu drevesu. S tem so dobili vizualno zadovoljive rezultate za do dvajset tisoč dreves. Upodabljanje je trajalo več ur za eno sliko.

Deussen in sodelavci [62] so prišli do spoznanj, da je za vizualizacijo mnogih rastlin potrebno rastline realistično postaviti po terenu, za rastline uporabiti kompakten predstavitveni model zaradi omejitev pomnilniških kapacitet in zmanjšati količino geometrije za upodobitev (z izdelavo primerkov). Realističnost postavitve rastlin na teren so zagotovili na dva možna načina: ročno ali s simulacijo. Pri prvem načinu, ročno, so rastline porazdelili na globalni ravni s sliko v velikosti ploskve pokrajine, ki je določila gostote rastlin. Iz gostot so dejanske položaje rastlin dobili s poltonskim algoritmom. Pred dokončno postavitvijo položajev rastlin so te prilagodili tako, da so rastline pomaknili v središča Voronojevih mnogokotnikov, ki so oklepali rastline. Kot drugi način porazdelitve rastlin na pokrajino, so lokacije dreves določili s simulacijo na individualni ravni z obravnavo **ekološke sosednosti** (*ecologic neighborhood*) posameznih rastlin. Izhajali so iz osnovnega fenomena porazdelitve rastlin, ki mu pravimo **samo-redčenje** (*self-thinning* [198]). Ta pove, da rastline na začetku rastejo brez medsebojnega oviranja, ob večji zgostitvi pa začnejo manjše rastline odmirati. Simulacijo so začeli z začetno populacijo rastlin, ki so jim dodelili naključno pozicijo in naključni polmer ekološke sosednosti. Vsaka rastlina je pripadala neki rastlinski vrsti, za katero so bili definirani parametri, kot so število dodanih novih rastlin na pokrajino ob vsakem simulacijskem koraku, največja velikost rastlin, povprečna hitrost rasti, verjetnost preživetja ob dominiranosti in potreba po vlagi. Tekmovalno sposobnost rastline so določili z zmnožkom

starosti rastline in parametra **živosti** (*vigour*). Živost rastline je bila izračunana za vsako rastlino posebej in je predstavljala zmnožek potrebe po vlagi in vsebnosti vlage na mestu rasti. Z naraščanjem simulacijskega časa so večali ekološke polmere rastlin in odstranjevali odmrle rastline. Rastline so lahko odmrle od starosti ali v primeru, ko so bile nadvladane. S simulacijo so dosegli samo-redčenje rastlin na terenu in porazdelitev vrst rastlin na mesta terena z ugodnimi pogoji. Model so Deussen in sodelavci [62] kasneje dopolnili z **rastjo rastlin blizu starševskih rastlin** (*clustering*, *clumping*, *underdispersion*) [131]. Hitro so namreč ugotovili, da je prejšnji model tvoril preveč enakomerno porazdeljene rastline, v naravi pa te rastejo precej po skupinah. Rast v skupinah so dosegli s **Hopkinsonovim indeksom** (*Hopkinson index*) [97]. Ta je definiran kot trenutna zgostitev rastlin na nekem mestu. Na osnovi tega indeksa so zgradili matriko verjetnosti postavitve rastline na določeno krpo površja. Rastline so postavili na površje glede na te verjetnosti, ob dodajanju nove rastline pa vrednosti v matriki ustrezno posodobili. Čeprav so z opisanim modelom dobili nekoliko boljše rezultate kot prej, pa so rezultati še vedno odsevali le malo dejanskih vplivov okolja.

Rastline so Deussen in sodelavci [62] vizualizirali z **izdelavo primerkov** (*instancing*) rastlinskih vrst. Vsako rastlinsko vrsto so predstavili le z enim modelom za primerek in le-tega so uporabili večkrat. Posamezne rastline so dobili tako, da so model primerka naključno zasukali okoli vertikalne osi. S tem so dosegli različno usmerjenost rastlin in posledično manj zaznavno podobnost med rastlinami pri upodobitvi. Njihov sistem za vizualizacijo je omogočal naravna okolja tudi do sto tisoč rastlin, vendar je za tako upodobitev potreboval do osem ur za eno sliko. Za predstavitev geometrije so Deussen in sodelavci [62] uporabili L-sisteme [186], rastline pa upodobili z algoritmom metanja žarka (*ray casting*) ali z algoritmom sledenja žarku (*ray tracing*). Pokrajino so sestavili s pomočjo lokalnih operatorjev nad ploskvijo površja, ki je vsebovala griče in potoke. Predogled scene je bil možen z OpenGL, vendar je realistično upodabljanje pokrajine z rastlinami trajalo več ur in ni bilo primerno za animacijo v realnem času. Na spletu je sicer možno najti več vizualizatorjev za hiter izris terena z OpenGL [181, 64], večina pa uporablja kakšno od razširitev OpenGL. Mrežo s 100×100 točkami ponavadi izrisujejo v realnem času, s približno 100 ali več sličicami na sekundo. Kvaliteta sicer ni tako dobra kot pri upodabljanju s sledenjem žarka, vendar je hitrost izrisa precej hitrejša. Tudi poraba pomnilnika je manjša pri upodabljanju z OpenGL, saj podano geometrijsko strukturo v programu lahko takoj zavržemo, ko smo jo prenesli na strežnik OpenGL, ta pa zaradi uporabe vmesnega pomnilnika geometrijsko strukturo po izrisu prav tako zavrže.

C.2 Model za simulacijo animiranega ekosistema

Predstavimo svoj pogled na problematiko ekološkega modeliranja in vlogo računalniške grafike pri realističnem prikazu naravnega procesa. V ta namen je pripravljen lasten ekološki model drevesnega ekosistema. Pri tem so pretvorjene znane zakonitosti iz ekologije in biologije v računalniške algoritme, ki omogočajo njihovo simulacijo v računalniku. V simulaciji evolucije ekosistema za vernejšo porazdelitev rastlin na terenu poleg tekmovanja med rastlinami za prostor in rast v bližini starševske rastline upoštevamo še lastnosti terena, kot so naklon, vlaga, osončenost in vetrovnost. S tem povečamo vernost porazdelitve rastlin in dosežemo rast rastlin v skupinah glede na ugodnosti življenjskih pogojev na določenem mestu. V ekosistemu na pokrajini vizualiziramo drevesa, ki jih generiramo iz modelirnika dreves. Slednji vezano na vizualizacijo pokrajine omogoča tudi animacijo in poenostavljanje geometrijske strukture. Animiramo rast dreves in zibanje v vetru, geometrijsko strukturo dreves pa poenostavljamo glede na oddaljenost od gledišča.

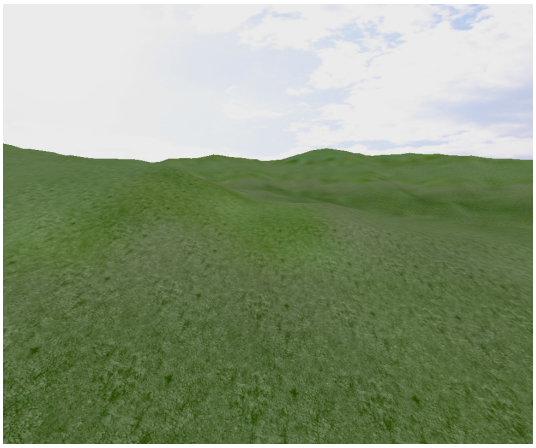
Drevesa razporejamo po terenu v odvisnosti od parametrov, ki vplivajo na njihovo rast. Na ta način simuliramo dogajanje v naravnem okolju. Rast dreves je v našem modelu odvisna od naslednjih parametrov: **nadmorske višine** (*altitude*) [150], **naklona** (*slope*), **vlage** (*humidity*), **osončenosti** (*sunniness*), **vetrovnosti** (*windiness*), **medsebojnega tekmovanja za prostor med rastlinami** (*competition for space*) [131], **rasti blizu starševskih rastlin** (*clustering*, *clumping*, *underdispersion*) [131, 47] in **umiranja rastlin zaradi staranja** (*dying with senescence*). Pri izračunu teh življenjskih pogojev si pomagamo s točkami terena. Pri naši študiji smo koristili prosto dostopni vizualizator, ki pa smo ga kasneje zamenjali z lastnim vizualizatorjem. Primer vizualizacije terena je prikazan na sliki C.1.

C.2.1 Terenski modeli

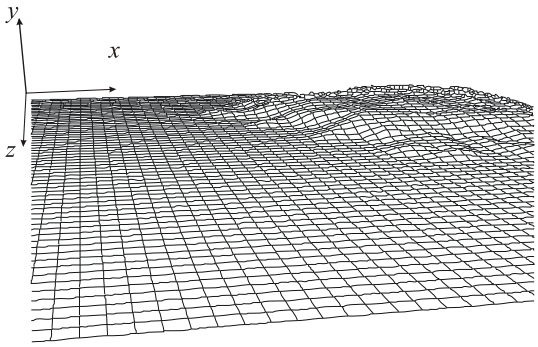
Površje je razdeljeno na 10.000 krp - četverokotnikov (100×100). Mreža, na kateri temelji predstavitev, je dodana terenu na sliki C.2 in je prikazana v perspektivni projekciji. Mreža četverokotnikov v ravnini xz je uniformna (slika C.3). Oglišča četverokotnikov v prostoru xyz določajo naslednje koordinate:

$$\mathbf{p}_{i,k} = \begin{bmatrix} x_i & y_{i,k} & z_k \end{bmatrix}^T, \qquad i,k \in [0,99]. \tag{C.1}$$

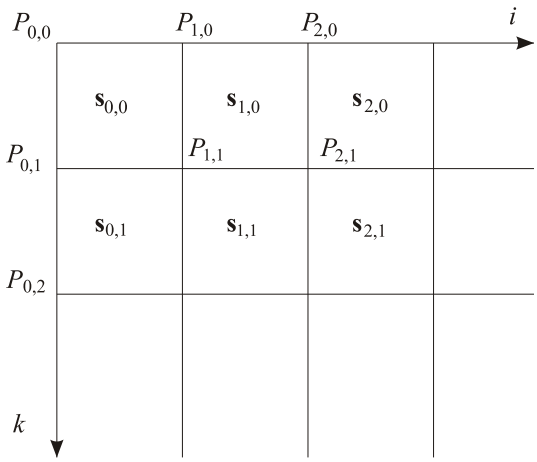
Četverokotniki so v prostoru zaradi različne nadmorske višine različnih velikosti. Označimo jih kot $\mathbf{s}_{i,k}$ $(i,k \in [0,99])$.



Slika C.1: Vizualizacija terena z vizualizatorjem za prikaz pokrajine.



Slika C.2: Mreža terena, sestavljena iz četverokotnikov.



Slika C.3: Mreža četverokotnikov v ravnini xz.

Točke terena je možno določiti z bitno sliko ali z nalaganjem DMR (digitalnega modela reliefa) [136]. V datoteki formata DMR so višinski podatki resničnega terena iz satelitov vzorčeni na 25 metrov. Koordinate oglišč terena so zapisane s po tremi vrednostmi, vsaka v svoji vrstici. Prva vrednost pomeni v našem simulatorju koordinato x, druga z in tretja y. Parametri, ki izhajajo neposredno iz razgibanosti terena, so nadmorska višina, naklon, vlaga, osončenost in vetrovnost. Te parametre dobimo z različnimi algoritmi pred pričetkom simulacije ekosistema in jih ne spreminjamo med tekom simulacije. Te parametre računamo za vsako krpo pokrajine posebej. V nadaljevanju bomo opisali algoritme, ki omogočajo izračun parametrov.

Nadmorska višina

Nadmorsko višino terena y normaliziramo po formuli:

$$\overline{y} = \frac{y - y_{\min}}{y_{\max} - y_{\min}}, \tag{C.2}$$

pri čemer je $y_{\min}$ minimalna nadmorska višina na terenu in $y_{\max}$ maksimalna nadmorska višina na terenu. Parameter višine terena $\overline{y}$ bomo uporabili pri odtekanju padavin in določanju vlažnosti zemlje.

Naklon

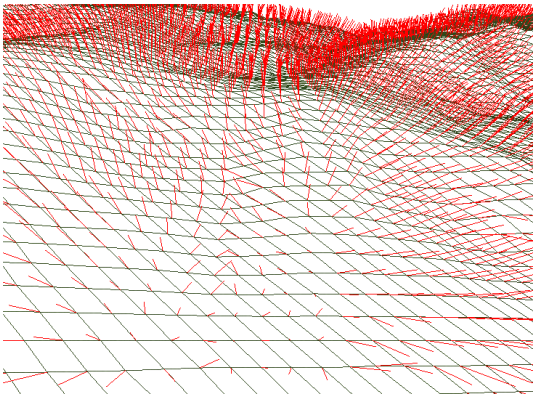
Naklon $s_{i,k}$ izračunamo iz komponente y enotskega normalnega vektorja:

$$s_{i,k} = 1 - \overline{n}_{y;i,k}. \tag{C.3}$$

Popolnoma ravni teren ima zato vrednost naklona 0, popolnoma navpični teren pa vrednost 1. Pri izračunu te enačbe potrebujemo normalo $\mathbf{n}_{i,k}$ četverokotnika (krpe) $\mathbf{s}_{ik}$. Le-to določimo z vektorskim produktom dveh vektorjev: prvi vektor povezuje obravnavano točko s sosednjo po dolžini terena, drugi vektor pa s sosednjo točko po širini terena:

$$\mathbf{n}_{i,k} = (\mathbf{p}_{i+1,k} - \mathbf{p}_{i,k}) \times (\mathbf{p}_{i,k+1} - \mathbf{p}_{i,k}) = \begin{bmatrix} n_{x;i,k} & n_{y;i,k} & n_{z;i,k} \end{bmatrix}^T. \tag{C.4}$$

Normale $\mathbf{n}_{i,k}$ normiramo in jih označimo kot $\overline{\mathbf{n}}_{i,k}$. Enotski vektorji normal terena so prikazani na sliki C.4.



Slika C.4: Enotski vektorji normal po terenu ($\overline{\mathbf{n}}_{i,k}$).

Vlaga

Na vsako krpo pade enaka količina padavin, te padavine pa ne ostanejo v celoti na določeni krpi, ampak se stekajo k sosednjim krpam (ali pritekajo od sosednjih krp) v odvisnosti od naklona krpe in vpojnosti tal. To pomeni, da vlago v krpi sestavljajo padavine, ki jih krpa neposredno vpije, in padavine, ki pritečejo iz višjih krp. V bližini tekoče ali mirujoče vode pa se vlagi v krpi doda tudi učinek kapilarnosti.

Krpe, ki sestavljajo teren, imajo v prostoru različno ploščino, vendar pa ima njihova projekcija na ravnino xz vedno enako ploščino. Privzamemo, da so padavine enakomerno porazdeljene po površini ravnine xz in da padajo pravokotno na ravnino xz. Po tej predpostavki zato prejme vsaka krpa (ne glede na velikost) enako količino padavin m_{climate}.

Vpojnost tal je funkcija strukture tal in vrste padavin (rahel dež, naliv). V naši simulaciji vpojnost tal trenutno opišemo z odstotkom padavin, ki jih tla vsrkajo.

Na začetku vsem krpam priredimo enako vrednost padavin m_{climate} (na primer $m_{\text{climate}} = 1$). Za vsako od krp upoštevamo še širjenje vlage okoli vodotokov in stoječih voda, tako da krpam vodotokov in stoječih voda dodamo velike količine padavin m_{w} (na primer $m_{\text{w}} = 10$), kot kaže enačba C.5. Krpe, ki predstavljajo vodotoke in stoječe vode, so tiste, ki imajo komponento $\overline{y}_{i,k}$ manjšo ali enako normalizirani višini nivoja vode $\overline{y}_{\text{w}} \in [0, 1]$, ta pa je nastavljivi parameter simulacije.

$$m_{i,k} := \begin{cases} m_{i,k} + m_{\text{w}}, & \overline{y}_{i,k} \leq \overline{y}_{\text{w}} \\ m_{i,k}, & \text{sicer} \end{cases}. \tag{C.5}$$

Krpe nato izbiramo po višini od najvišje proti nižjim. Denimo, da je izbrana krpa $\mathbf{s}_{i,k}$. Najprej določimo število strani odtekanj te krpe $n_{\text{neigh};i,k}$, ki ga dobimo tako, da preštejemo sosednje krpe (0 do 4), ki imajo nižjo središčno točko.

Pri odtekanju padavin upoštevamo še moč odtekanja k_{m}, ki je uporabniško nastavljiv parameter. Delež odlitja padavin na eno sosednjo krpo je za trenutno krpo določen s formulo:

$$\Delta m_{i,k} = \begin{cases} 0, & n_{\mathrm{neigh};i,k} = 0 \\ \dfrac{s_{i,k} k_{\mathrm{m}}}{n_{\mathrm{neigh};i,k}}, & n_{\mathrm{neigh};i,k} = 1, 2, 3, 4 \end{cases}. \tag{C.6}$$

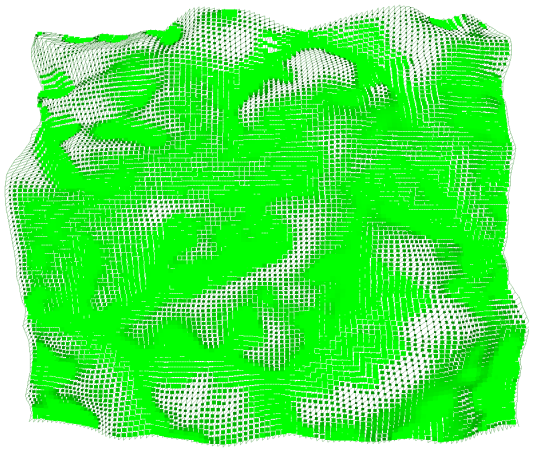
To pomeni, da iz bolj strme krpe padavine odtekajo močneje, iz ravne krpe pa padavine ne odtekajo. Ta delež prištejemo sosednjim krpam, ki imajo nižjo središčno točko:

$$m_{i\pm1,k\pm1} := m_{i\pm1,k\pm1} + \Delta m_{i,k}. \tag{C.7}$$

Vse odlite deleže pa trenutni krpi vselej tudi odštejemo:

$$m_{i,k} := m_{i,k} - n_{\mathrm{neigh};i,k} \Delta m_{i,k}. \tag{C.8}$$

Celotni izračun vlage na koncu N_{mblur}-krat povprečimo (npr. $N_{\mathrm{mblur}} = 10$), s čimer zmehčamo izstopajoče vrednosti in dobimo zvezno širjenje vlage po prsti. Povprečenje za neko krpo v enem koraku opravimo tako, da zanjo shranimo povprečno količino vlage na njej in štirih ostalih obkrožajočih krpah. Splošneje rečeno, nad podatki izvedemo digitalni filter nizkega sita (tekoče povprečenje). Postopek za izračun vlage je zapisan z algoritmom 16. Izračunana vlažnost v krpah, ki jo dobimo v programu, je prikazana z velikostjo kvadratov na sliki C.5.



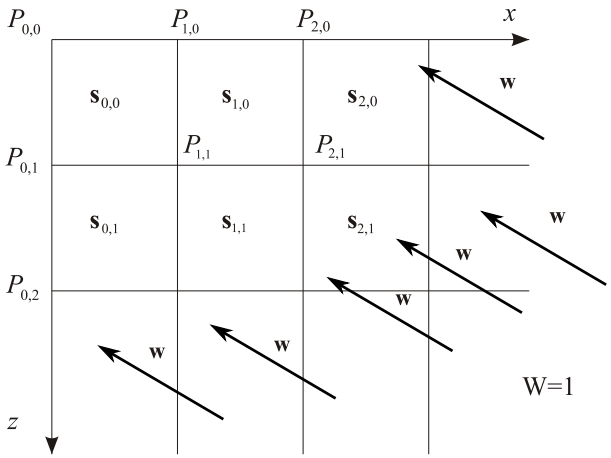
Slika C.5: Izračunana vlažnost terena s simulacijo odtekanja padavin. Območja z več zelene barve so bolj vlažna.

Algoritem 16 Izračun padavin za teren.

Vhod: $i_{\max}$, $k_{\max}$ - dimenzije terena; $\mathbf{c}_{i,k}$ - središčne točke krp terena; N - število krp terena

Izhod: matrika elementov $m_{i,k}$ - vlaga za vsako krpo

for $i = 0$ **to** $i_{\max} - 1$ **do**
  for $k = 0$ **to** $k_{\max} - 1$ **do**
    $m_{i,k} := m_{\text{climate}}$;
    if $\overline{y}_{i,k} \leq \overline{y}_{\text{w}}$ **then**
      $m_{i,k} := m_{i,k} + m_{\text{w}}$; {dodamo vlago krpam, ki ustrezajo vodotokom in stoječim vodam}
    end if
  end for
end for
krpe uredi nenaraščajoče po višini $(\mathbf{c}_{i,k}, N)$; {dobimo npr. seznam (34, 13, 17, ...)}
for $j = 0$ **to** $N - 1$ **do**
  $i, k :=$ dobi indeks i, k iz indeksa j; {dobimo $0 \to 34$, $1 \to 13$, $2 \to 17$, ...}
  $n_{\text{neigh};i,k} :=$ določi število strani odtekanj iz krpe $\mathbf{s}_{i,k}$;
  $\Delta m_{i,k} := m_{i,k} \frac{s_{i,k} k_{\text{m}}}{n_{\text{neigh};i,k}}$; {izračunamo delež odlitja za eno krpo}
  $m_{i\pm1,k\pm1} := m_{i\pm1,k\pm1} + \Delta m_{i,k}$; {sosednjim krpam prištejemo delež padavin}
  $m_{i,k} := m_{i,k} - n_{\text{neigh};i,k} \Delta m_{i,k}$; {trenutni krpi odštejemo deleže odteklih padavin}
end for
for $j = 0$ **to** $N_{\text{mblur}} - 1$ **do**
  $m_{i,k} := \frac{m_{i,k} + m_{i-1,k} + m_{i,k+1} + m_{i+1,k} + m_{i,k+1}}{5}$; {povečamo vlago krpam ob vodotokih in stoječih vodah}
end for



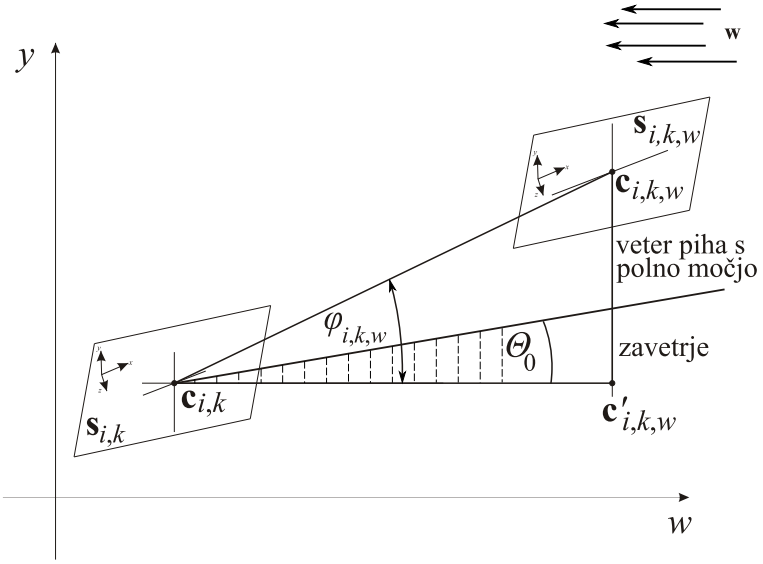
Slika C.6: Pihanje vetra v ravnini xz čez cel teren v isti smeri. Jakost vetra W je konstantna za ves teren, prav tako pa je konstantna tudi smer vetra.

Vetrovnost

Vetrovnost $w_{i,k}$ v krpi $\mathbf{s}_{i,k}$ določa izpostavljenost te krpe vetru. Ta piha konstantno z jakostjo $W = 1$ s smerjo vektorja $\mathbf{w}$ v ravnini xz, kot prikazuje slika C.6.

Če so sosednje krpe v smeri $-\mathbf{w}$ višje ležeče od obravnavane, je krpa v zavetrju. Količino vetrovnosti $w_{i,k}$ določimo s pomočjo kota zavetrja $\varphi_{i,k,w} = \measuredangle(\mathbf{c}_{i,k,w}, \mathbf{c}_{i,k}, \mathbf{c}'_{i,k,w})$, pri čemer je točka $\mathbf{c}_{i,k,w}$ središčna točka krpe v smeri proti vetru, $\mathbf{c}_{i,k} = \begin{bmatrix} c_{x;i,k} & c_{y;i,k} & c_{z;i,k} \end{bmatrix}^T$ središčna točka obravnavane krpe in $\mathbf{c}'_{i,k,w}$ projekcija točke $\mathbf{c}_{i,k,w}$ na ravnino $y = c_{y;i,k}$ (slika C.7). Kot zavetrja izračunamo kot:

$$\varphi_{i,k,w} = \arctan \frac{c_{y;i,k,w} - c_{y;i,k}}{\|\mathbf{c}'_{i,k,w} - \mathbf{c}_{i,k}\|}. \tag{C.9}$$

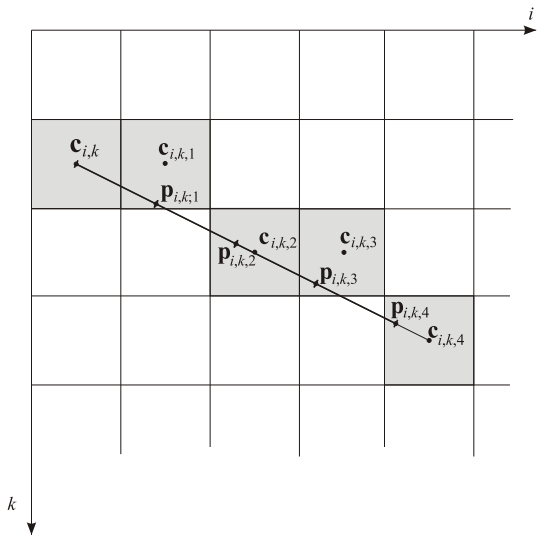


Slika C.7: Kot zavetrja $\varphi_{i,k,w}$ med središčnima točkama $\mathbf{c}_{i,k}$ in $\mathbf{c}_{i,k,w}$ krp $\mathbf{s}_{i,k}$ in $\mathbf{s}_{i,k,w}$. Krpa $\mathbf{s}_{i,k,w}$ ustreza w-temu vzorcu — ti so na sliki C.8 označeni s črticami.

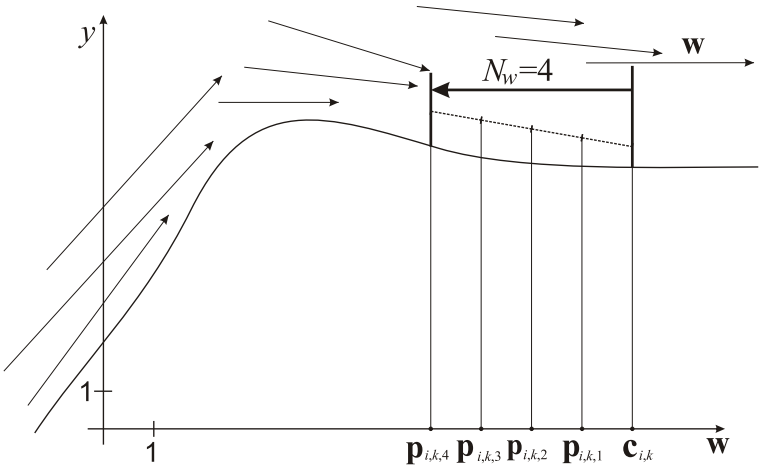
Zavetrje v krpi določimo iz najmanjšega kota zavetrja ($\varphi_{i,k} = \min_{\mathbf{w}} \{\varphi_{i,k,w}\}$). Postopek sledenja vektorju proti vetru je prikazan na sliki C.8.

Da dokončno določimo vetrovnost $w_{i,k}$, upoštevamo le še spuščanje vetra, ki ga postopoma modeliramo na dva načina, kot to prikazuje slika C.9.

Prvi način je z največjim radijem N_{w}, ki omejuje dolžino potovanja v nasprotni smeri vetra in je določen s številom krp (običajno 10 [231]). Drugi način je s pomočjo praga $\Theta_0 \in [0, \frac{\pi}{2}]$ (npr. $\frac{\pi}{4}$) za kot $\varphi_{i,k}$, ko zavetrje popolnoma izgine, ker se veter usmerja navzdol. V preseku kot Θ_0 prikazuje slika C.7. Dokler je $\varphi_{i,k}$ manjši od Θ_0,



Slika C.8: Sledenje proti vetru iz točke $c_{i,k}$ za $N_{\mathrm{w}} = 4$ vzorcev. Smer potovanja je določena z $-\mathbf{w}$, dolžina koraka je enaka dimenziji krpe K. Iz vzorcev (črtice) dobimo pripadajoče krpe in njihove središčne točke.



Slika C.9: Spuščanje vetra. Veter se lahko po nekaj krpah toliko spusti, da že piha s polno močjo.

vetrovnost $w_{i,k}$ narašča linearno, kasneje pa je ta vrednost enaka 1:

$$w_{i,k} = \begin{cases} \varphi_{i,k}, & \varphi_{i,k} \leq \Theta_0 \\ 1, & \text{sicer} \end{cases}. \tag{C.10}$$

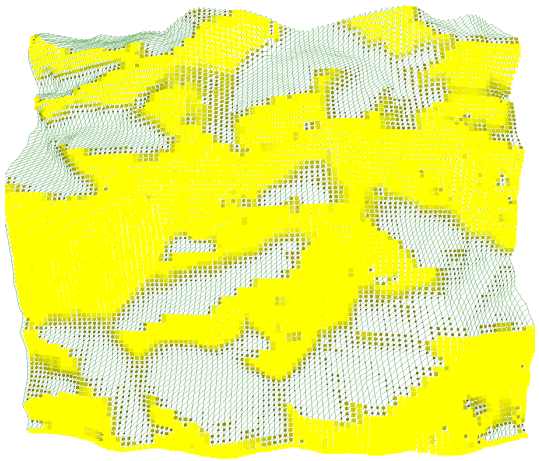
Opisan postopek je prikazan kot algoritem 17.

Algoritem 17 Izračun vetrovnosti za teren.

Vhod: $i_{\max}$ - število krp terena po osi x; $k_{\max}$ - število krp terena po osi z; N_{w} - število krp v smeri proti vetru
Izhod: matrika elementov $w_{i,k}$ - vetrovnost za vsako krpo

```
for i = 0 to i_max − 1 do
  for k = 0 to k_max − 1 do
    φ_{i,k} := 0;
    for w = 0 to N_w − 1 do
      φ_{i,k,w} = arctan (c_{y;i,k,w} − c_{y;i,k}) / ||c'_{i,k,w} − c_{i,k}||;
      φ_{i,k} := min(φ_{i,k,w}, φ_{i,k}); {manjši ko je kot zavetrja, bolj piha veter}
    end for
    if φ_{i,k} ≤ S_0 then
      w_{i,k} := φ_{i,k};
    else
      w_{i,k} := 1;
    end if
  end for
end for
```

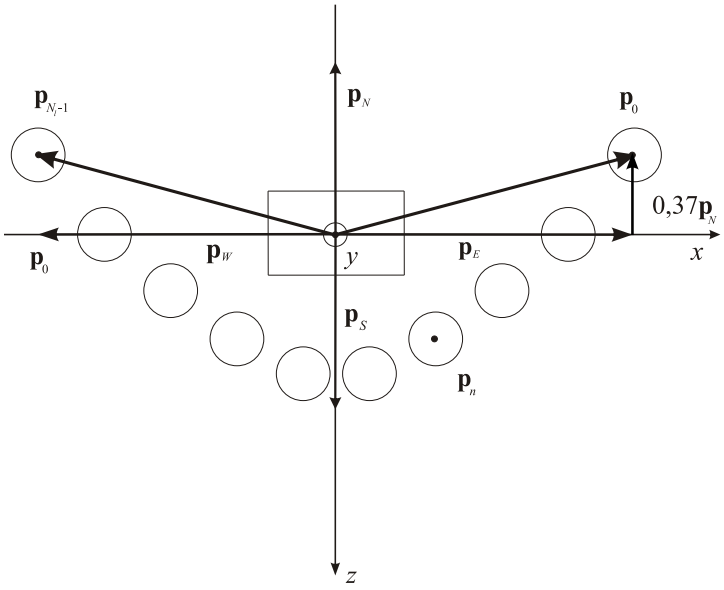
Izračunana vetrovnost, ki jo izračuna program, je prikazana na sliki C.10.



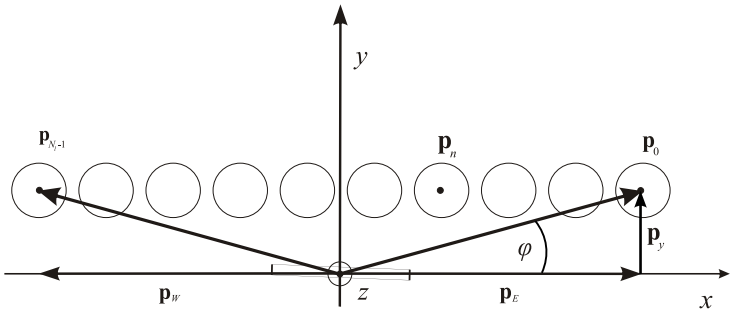
Slika C.10: Vpliv vetra na terenu. Območja z več rumene barve so bolj vetrovna.

Osončenost

Na terenu določimo normirane vektorje v smeri strani neba $\mathbf{p}_N$, $\mathbf{p}_S$, $\mathbf{p}_E$, $\mathbf{p}_W$ in simuliramo potovanje sonca[1] od vzhoda proti zahodu [252]. Ker je simulator prirejen za severno zemeljsko poloblo, sonce nekoliko nagnemo proti jugu, tako da so južni deli terena bolj osvetljeni od severnih. Skrajne položaje sonca $\mathbf{p}_0$ in $\mathbf{p}_{N_1-1}$ ter gibanje sonca $\mathbf{p}_n$ med njima vidimo na slikah C.11 in C.12.



Slika C.11: Vzorci položajev sonca pri izračunu osvetljenosti za neko krpo $\mathbf{s}_{i,k}$.

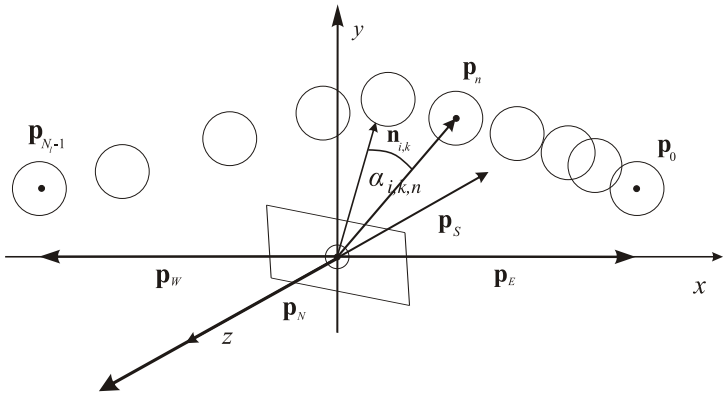


Slika C.12: Vpadni kot sončne svetlobe, ki odreja osončenost krpe. S spreminjanjem točk vzorčenja se spreminja tudi ta kot.

Vektorja $\mathbf{p}_0$ in $\mathbf{p}_{N_1-1}$ določata naslednji enačbi:

$$\mathbf{p}_0 = \mathbf{p}_E + 0,37\mathbf{p}_N, \tag{C.11}$$

[1]Za pomoč pri razlagi astronomskih koordinatnih sistemov in izpeljavi kota α_M se zahvaljujemo doc. dr. Vladimirju Grubelniku.



Slika C.13: Vzorci položajev sonca pri izračunu osvetljenosti za neko krpo $\mathbf{s}_{i,k}$, v prostoru.

$$\mathbf{p}_{N_l-1} = \mathbf{p}_W + 0,37\mathbf{p}_N, \tag{C.12}$$

kjer vektor $0,37\mathbf{p}_N$ pomeni, da kot vzhoda in zahoda pomaknemo proti severu (za izpeljavo konstant glej dodatek C.2.1).

Osončenost računamo v N_l položajih sonca, tako da upoštevamo še spremembo deklinacijskega kota srednjega vzorca $\frac{N_l-1}{2}$. Vektor sonca v najvišji legi je na 46. vzporedniku (kot deklinacije je 46°) zaradi vpadnega kota $\alpha_{i,k,\frac{N_l-1}{2}} = 90° - 46° = 44°$ enak $\mathbf{p}_{\frac{N_l-1}{2}} = (0,\ \sin 44°,\ \cos 44°)$, sonce pa je po sončni uri ob 12:00 natanko med vzhodom in zahodom. Opisan izračun podaja naslednja enačba:

$$\mathbf{p}_n = \begin{cases} \mathbf{p}_0 + \frac{2n}{N_l-1}(\mathbf{p}_{\frac{N_l-1}{2}} - \mathbf{p}_0), & n = 0, 1, ..., \frac{N_l-1}{2} \\ \mathbf{p}_0 + \frac{2(n-\frac{1}{2})}{N_l-1}(\mathbf{p}_{N_l-1} - \mathbf{p}_{\frac{N_l-1}{2}}), & n = \frac{N_l-1}{2} + 1, ..., N_l - 1 \end{cases}. \tag{C.13}$$

Dobljeni vektor $\mathbf{p}_n$ normaliziramo in izračunamo kosinus vpadnega kota, t.j. kota $\alpha_{i,k,n}$ med enotsko normalo $\overline{\mathbf{n}}_{i,k}$ terena in vektorjem proti soncu $\mathbf{p}_n$ (sliki C.14 in C.13):

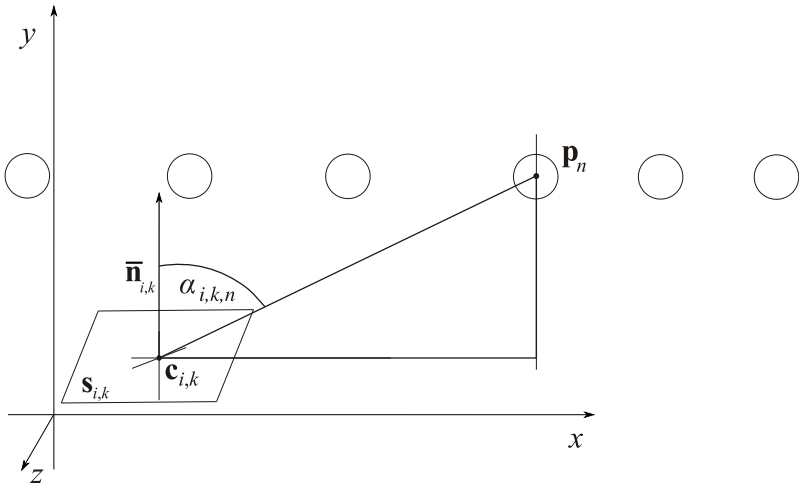
$$\cos\alpha_{i,k,n} = \overline{\mathbf{n}}_{i,k} \cdot \frac{\mathbf{p}_n}{\parallel \mathbf{p}_n \parallel}. \tag{C.14}$$

Osončenost krpe $l_{\text{basic};i,k,n}$, če je sonce v položaju $\mathbf{p}_n$, določa enačba:

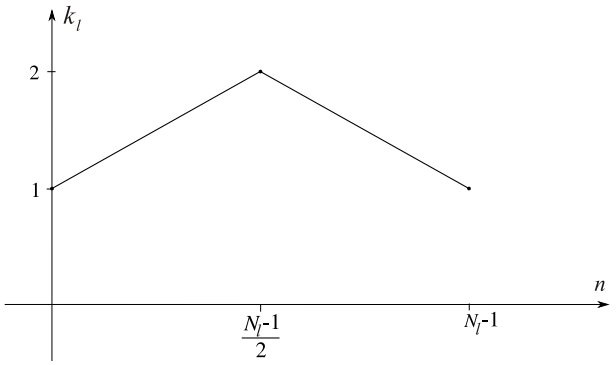
$$l_{\text{basic};i,k,n} = \begin{cases} \cos\alpha_{i,k,n}, & 0 \le \alpha_{i,k,n} \le \frac{\pi}{2} \\ l_{\text{ambient}}, & \text{sicer} \end{cases}. \tag{C.15}$$

Vrednost osončenosti $l_{\text{basic};i,k,n}$ smo utežili s faktorjem k_l, ki ga prikazuje slika C.15, kjer predpostavljamo, da je N_l liho število. S tem damo večjo težo položajem, ki so sredi dneva, saj je ozračje bolj segreto in ima sonce večji vpliv. Nove vrednosti osončenosti so tako:

$$l_{i,k,n} = \begin{cases} l_{\text{basic};i,k,n}\left(1 + \frac{2n}{N_l-1}\right), & 0 \le n \le \frac{N_l-1}{2} \\ l_{\text{basic};i,k,n}\left(3 - \frac{2n}{N_l-1}\right), & \frac{N_l-1}{2} < n \le N_l - 1 \end{cases}. \tag{C.16}$$



Slika C.14: Vpadni kot sončne svetlobe, ki odreja osončenost krpe. S spreminjanjem točk vzorčenja se spreminja tudi ta kot.



Slika C.15: Korekcijski faktor k_l osončenosti sredi dneva, saj je ozračje bolj segreto in ima sonce večji vpliv.

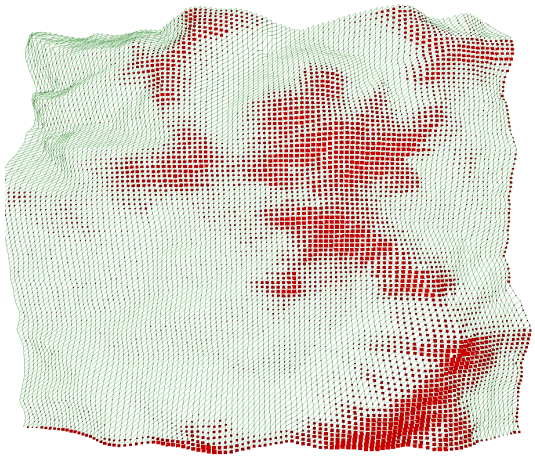
Pri izračunu osončenosti upoštevamo še senčnost krp, ki jih izračunamo podobno kot pri algoritmu za vetrovnost (algoritem 17). Če je krpa v senci, osončenost iz trenutnega vzorca zmanjšamo glede na nastavljivi koeficient $l_{\text{ambient}} \in (0, 1)$ za prisotnost razpršene svetlobe (npr. $l_{\text{ambient}} = 0,2$), sicer ga pustimo nespremenjenega.

Pri sledenju v smeri proti vzorcu položaja sonca $\mathbf{p}_n$ tako preverjamo, ali obstaja kakšna krpa, ki zastira sončevo svetlobo. To ugotovimo s primerjavo kotov od srednje točke trenutne krpe $\mathbf{c}_{i,k}$ do vzorca sonca in od iste točke do n-te srednje točke $\mathbf{c}_{i,k;n}$ krpe na sledi proti vzorcu sonca. Omenjen kot izračunamo enako kot pri algoritmu za veter, po enačbi C.9. Če je izračunan kot večji od kota med trenutno krpo $\mathbf{c}_{i,k}$ in vzorcem položaja sonca, je trenutna krpa v senci in sledenje končamo. Končamo tudi takrat, če pridemo na rob terena in tedaj krpa ni v senci.

Osončenost krpe $\mathbf{s}_{i,k}$ predstavlja vsota osončenosti v vseh položajih sonca:

$$l_{i,k} = \frac{\sum_{n=0}^{N_\text{l}-1} l_{i,k,n}}{N_\text{l}}. \tag{C.17}$$

Algoritem za izračun osončenosti kaže algoritem 18. Slika C.16 kaže vrednost osončenosti s pomočjo velikosti kvadratov na krpah.



Slika C.16: Vpliv sonca izračunan na realnih podatkih terena. Območja z več rdeče barve so bolj osončena.

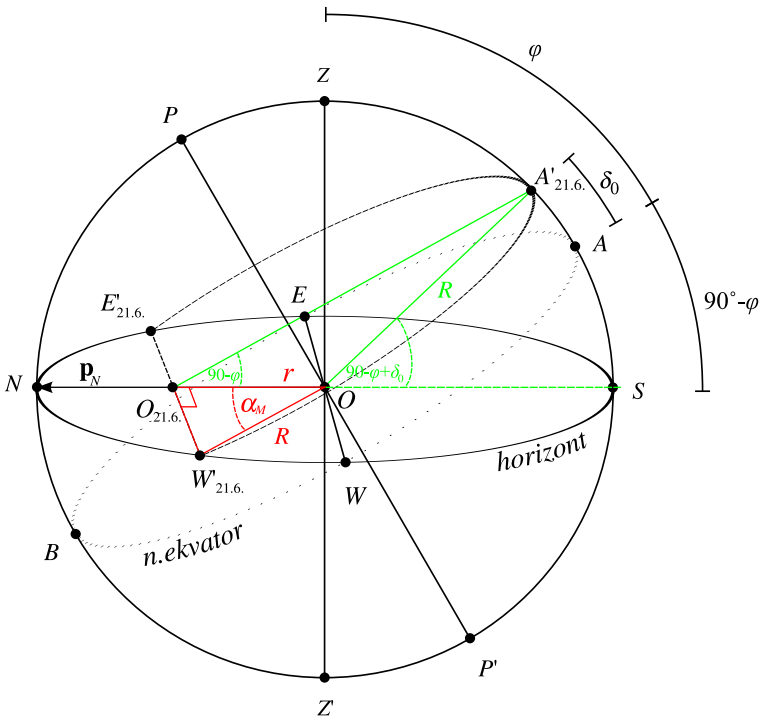
Izpeljimo povprečen kot osončenja v poletnem času med enakonočjema. V modelu osončenja vzamemo povprečni kot vzhoda in zahoda sonca skozi pomladansko in poletno obdobje (od pomladanskega enakonočja 21. marca do jesenskega enakonočja 23. septembra), da pri osončenju terena izračunamo vektor $\mathbf{p}_0$. Ta kot znaša približno $\alpha = 21,98°$ proti severu, zato vektorju položaja sonca $\mathbf{p}_0$ prištejemo približno $0,37\mathbf{p}_N$. Izpeljava sledi.

Algoritem 18 Izračun osvetljenosti za teren.

Vhod: $i_{\max}$ - število krp terena po osi x; $k_{\max}$ - število krp terena po osi z; N_l - število vzorcev za položaje sonca; $\mathbf{p}_N$, $\mathbf{p}_S$, $\mathbf{p}_E$, $\mathbf{p}_W$ - vektorji iz terena v smeri strani neba

Izhod: matrika elementov $l_{i,k}$ - osvetljenost za vsako krpo

$\mathbf{p}_0 := \mathbf{p}_E + 0,37\mathbf{p}_N$; {določimo skrajne lege sonca}
$\mathbf{p}_{N_\text{l}-1} := \mathbf{p}_W + 0,37\mathbf{p}_N$;
$\mathbf{p}_{\frac{N_\text{l}-1}{2}} = (0,\ \sin 44°,\ \cos 44°)$;
for $i = 0$ **to** $i_{\max} - 1$ **do**
  for $k = 0$ **to** $k_{\max} - 1$ **do**
    $l_{i,k} := 0$; {inicializacija celotne osvetljenosti za krpo}
    for $n = 0$ **to** $N_\text{l} - 1$ **do**
      if $n \leq N_\text{l} - 1$ **then**
        $\mathbf{p}_n = \mathbf{p}_0 + \frac{2n}{N_\text{l}-1}(\mathbf{p}_{\frac{N_\text{l}-1}{2}} - \mathbf{p}_0)$; {položaj sonca pri dopoldanskem vzorcu}
      else
        $\mathbf{p}_n = \mathbf{p}_0 + \frac{2(n-\frac{1}{2})}{N_\text{l}-1}(\mathbf{p}_{N_\text{l}-1} - \mathbf{p}_{\frac{N_\text{l}-1}{2}})$; {položaj sonca pri popoldanskem vzorcu}
      end if
      $\cos\alpha_{i,k,n} = \overline{\mathbf{n}}_{i,k} \cdot \frac{\mathbf{p}_n}{\|\mathbf{p}_n\|}$; {kot osvetljenosti vzorca}
      if $\cos\alpha_{i,k,n} \geq 0$ **then**
        $l_{\text{basic};i,k,n} = \cos\alpha_{i,k,n}$; {osvetljenost direktno osončenega vzorca}
        if $0 < l < \frac{N_\text{l}-1}{2}$ **then**
          $l_{i,k,n} = l_{\text{basic};i,k,n}\left(1 + \frac{2n}{N_\text{l}-1}\right)$; {jakost osvetljenosti vzorca čez dan}
        else
          $l_{i,k,n} = l_{\text{basic};i,k,n}\left(3 - \frac{2n}{N_\text{l}-1}\right)$; {jakost osvetljenosti vzorca na robovih dneva}
        end if
        if ni v senci($\mathbf{p}_n$) **then**
          $l_{i,k} := l_{i,k} + \frac{l_{i,k,n}}{N_\text{l}}$; {akumulacija celotne osvetljenosti za krpo}
        else
          $l_{i,k} := l_{i,k} + \frac{l_{i,k,n} l_{\text{ambient}}}{N_\text{l}}$; {upoštevamo razpršeno svetlobo v senci}
        end if
      end if
    end for
  end for
end for



Slika C.17: Točke in krogi nebesne krogle (sfere).

Znano je, da 21. marca in 23. septembra sonce vzide na vzhodu in zaide na zahodu. Takrat namreč sonce v ekvatorju sije pravokotno na Zemljo in nastopi enakonočje, saj sta dan in noč enako dolga. Od enega do drugega enakonočja v poletnem času na severni polobli sonce vzhaja nekoliko proti severu in zahaja prav tako nekoliko proti severu. To pomeni, da sonce čez dan po svoji poti na nebu opravi več kot 180°. Da bi dobili povprečen kot, koliko proti severu sonce vzhaja, moramo najprej določiti njegov ekstrem α_{M}, to je 21. junija, na dan poletnega solsticija.

Na sliki C.17 vidimo geometrijo nebesne krogle z nekaterimi točkami in krogi, kot bi jo lahko poenostavljeno modelirali. O označuje središče nebesne polkrogle, kjer je opazovalec. Z označuje zenit (nadglavišče), Z' nadir (podnožišče), P severni nebesni pol (tečaj), P' južni nebesni pol, E, W, S in N pa vzhod, zahod, sever, jug. Premico skozi ZO imenujemo navpičnica, premico skozi PP' nebesna os. Krivulja skozi EAW določa nebesni ekvator, skozi ESW pa horizont (obzorje). Krivulja NPZ določa nebesni meridian (poldnevnik). φ označuje geografsko širino opazovalca. δ_0 označuje odmik v stopinjah od enakonočja in je v času poletnega solsticija enak $23,5°$. Oznaki $E'_{21.6.}$ in $W'_{21.6.}$ označujeta točki vzhoda in zahoda sonca na dan poletnega solsticija na točki z geografsko širino $\varphi - \delta_0$, krivulja skozi njiju in $A'_{21.6.}$ pa leži v ravnini, vzporedni EAW. Z rdečo barvo poudarjen trikotnik nam da iskani kot α_{M}, iz katerega kasneje

izračunamo povprečni α. Kot α_{M} je napet med vektorjem dolžine polmera zemlje R, v smeri do zahoda sonca na dan poletnega solsticija in vektorjem v smeri proti severu, dolžine r.

Slovenija leži na 46. vzporedniku ($\varphi = 46°$), kar pomeni, da je v času poletnega solsticija zaradi nagiba $\delta_0 = 23,5°$ kot sonca enak $90° - 46° + 23,5° = 67,5°$ (kot $\measuredangle A_{21.6.}OS$ na sliki C.17). Iz kota $\measuredangle A_{21.6.}O_{21.6.}S = 44°$ pa lahko po sinusnem izreku izračunamo razmerje med dolžinama r in R. Razmerje med polmerom Zemlje R in delčkom polmera, ki se seka s horizontom (ESW: vzhod-jug-zahod), znaša:

$$\frac{r}{\sin \measuredangle O_{21.6.}A_{21.6.}O} = \frac{R}{\sin \measuredangle OO_{21.6.}A_{21.6.}}, \tag{C.18}$$

$$\frac{r}{\sin\left(180° - (90° - \varphi) - (180° - (90° - \varphi + \delta_0))\right)} = \frac{R}{\sin\left(90° - \varphi\right)}, \tag{C.19}$$

$$\frac{r}{\sin 23,5°} = \frac{R}{\sin 44°}, \qquad \frac{r}{R} = \frac{\sin 23,5°}{\sin 44°}. \tag{C.20}$$

Kot α_{M} v spodnjem pravokotnem trikotniku med stranico r in hipotenuzo R izračunamo:

$$\sin \alpha_{\mathrm{M}} = \frac{r}{R} = \sin^{-1} \frac{r}{R} \approx \sin^{-1} \frac{\sin 23,5°}{\sin 44°} = \sin^{-1} 0,574 = 35,031. \tag{C.21}$$

Ker se Zemlja okoli sonca vrti po elipsi, ki jo je za naš izračun možno aproksimirati s krožnico, se α_{M} s časom spreminja po sinusnem zakonu, če si za $t = 0$ izberemo pomladno enakonočje.

Za izračun povprečnega kota α čez poletje moramo izračunati polovico celoletne sinusne periode s $t_0 = 365,25$ dni. Kot $\alpha = 0$ je v času spomladanskega enakonočja ($t = 0$), v času poletnega solsticija $t = \frac{t_0}{4}$ je $\alpha = \alpha_{\mathrm{M}}$, v času jesenskega enakonočja pri $t = \frac{t_0}{2}$ pa spet $\alpha = 0$. Ker je sinusoida simetrična, moramo izračunati samo povprečje na prvi četrtini. Najprej pa izračunamo integral, da bi kasneje izračunali še povprečje:

$$\omega = \frac{2\pi}{t_0}, \qquad \alpha_{sum} = \int_0^{\frac{t_0}{2}} \alpha_{\mathrm{M}} \sin(\omega t) dt = 2\alpha_{\mathrm{M}} \int_0^{\frac{t_0}{4}} \sin(\omega t) dt \tag{C.22}$$

$$\alpha_{sum} = 2\alpha_{\mathrm{M}}(-\cos(\omega t)|_{\frac{t_0}{4}} - (-\cos(\omega t)|_0)) = 2\alpha_{\mathrm{M}}(0 + 1) \approx 70°. \tag{C.23}$$

Da bi dobili povprečje, rezultat integrala delimo s časom, skozi katerega se vrednosti kumulirajo, t.j. pol leta ($\Delta t = \frac{t_0}{2}$):

$$\alpha = \frac{\alpha_{sum}}{\frac{t_0}{2}} \approx \frac{70°}{\frac{365,25}{2}} = 0,383 = 21,980°. \tag{C.24}$$

Za izračun položaja sonca $\mathbf{p}_0$ moramo prišteti vektor proti severu, ki je dolžine sinusa kota α:

$$\mathbf{p}_0 = \mathbf{p}_E + \sin(\alpha)\mathbf{p}_N, \qquad \mathbf{p}_0 = \mathbf{p}_E + 0,374\mathbf{p}_N. \tag{C.25}$$

C.2.2 Evolucijski model posameznega drevesa

Drevesa na pokrajino dodajamo postopoma. Dodajanje rastlin rastlinskim vrstam lahko izvajamo zaporedoma, tako da za vsako vrsto dodamo določeno število $\Delta N_{g,s}$ dreves na teren. Zaenkrat povejmo le, da je to število določeno za posamezno vrsto posebej (iz zbiranja semen na terenu), saj je natančneje definirano kasneje pri postopku rasti rastlin (razdelek C.2.2). Drevesa nameščamo z upoštevanjem koncepta o rasti v bližini starševskih rastlin. Opisan postopek je predstavljen z algoritmom 19.

Algoritem 19 Osnovni algoritem za sajenje novih rastlin.

Vhod: N_s - število rastlinskih vrst; $\Delta N_{g,s}$ - število novih rastlin v tej generaciji pri s-ti rastlinski vrsti ($s = 1, ...N_s$); p_s - verjetnost rasti nove rastline blizu svoje vrste za vsako vrsto posebej

Izhod: nove rastline $P_{s,n}$ so dodane rastlinskim vrstam in na teren

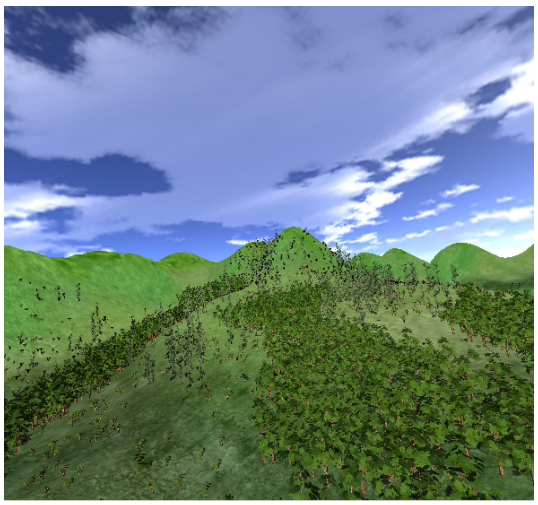
```
for s = 0 to N_s − 1 do
  for n = 0 to ΔN_{g,s} − 1 do
    if izpolni(p_s) then
      P_{parent;s,n} := poišči bližnjo rastlino(s); {izberemo obstoječo rastlino iz rastlinske vrste s}
      s_{s,n} := vrni naključno sosednjo krpo(P_{parent;s,n}); {naključje lahko sicer vrne tudi isto krpo}
      p_{s,n} := naključna točka na krpi(s_{s,n}); {dokončno določimo položaj nove rastline}
    else
      s_{s,n} := vrni naključno krpo();
      p_{s,n} := naključna točka na krpi(s_{s,n});
    end if
    if položaj p_{s,n} ni v vodi then
      P_{s,n} := tvori novo rastlino(p_{s,n}); {v podatkovno strukturo rastline shranimo položaj p_{s,n} in ostale pogoje terena na tem položaju}
    end if
  end for
end for
```

Kopičenje rastlin v bližini starševskih rastlin

Algoritem 19 se ne obnese dobro pri velikem številu rastlin (npr. nad 5.000 rastlin). Struktura za hranjenje rastlin v rastlinski vrsti je linearno povezan seznam $P_{s,p}$ in algoritem bi pri iskanju naključne bližnje rastline vsakokrat znova preiskal obstoječe rastline za vrsto. Zahtevnost algoritma je tako odvisna od števila rastlin in je $O(n^2)$. Zato raje uporabimo prilagojen algoritem 20, ki z uporabo **referenčnega seznama**

porazdelitve rastlin po krpah preišče sezname bližnjih rastlin posameznih vrst rastlin le enkrat na simulacijski korak in zmanjša zahtevnost algoritma dodajanja rastlin iz $O(n^2)$ na $O(n)$.

Iskanje bližnjih rastlin izvedemo na naslednji način. Indekse omenjenega referenčnega seznama označimo z $I_{s,n}$ in jih tvorimo za vsako rastlinsko vrsto posebej. Seznam osvežujemo tako, da se za novo nastale rastline na simulacijskem koraku naključno odločimo, ali rastlina raste blizu druge rastline ali kjerkoli drugje na terenu. Če se odločimo za slednjo možnost, povečamo števec rastlin, ki jih dodamo kjerkoli (označimo ga z A_s) in nadaljujemo. Pri prvi možnosti pa naključno izberemo indeks rastline iste rastlinske vrste kot je trenutno dodajana rastlina, blizu katere naj zraste nova rastlina in ga shranimo v seznam elementov $\{I_{s,n}\}$. Po končani gradnji seznama elementov $\{I_{s,n}\}$ tega nepadajoče uredimo glede na hranjene indekse $I_{s,n}$. Ker je tedaj seznam urejen, se lahko skozi seznam rastlin v njeni vrsti sprehodimo linearno od 0 do dolžine seznama $\{I_{s,n}\}$ in zgolj primerjamo indekse iz $I_{s,n}$. Če je tekoči indeks enak indeksu $I_{s,n}$, je trenutna rastlina starš novi rastlini, ki jo kreiramo blizu. Novo rastlino tako posadimo v majhnem naključnem odmiku od obstoječe rastline. Opisana izboljšava je zapisana v algoritmu 20. Primer izrazite rasti blizu svoje vrste in prevlado na tem območju zaradi ugodnih pogojev prikazuje slika C.18.



Slika C.18: Rastline rastejo blizu starševskih rastlin.

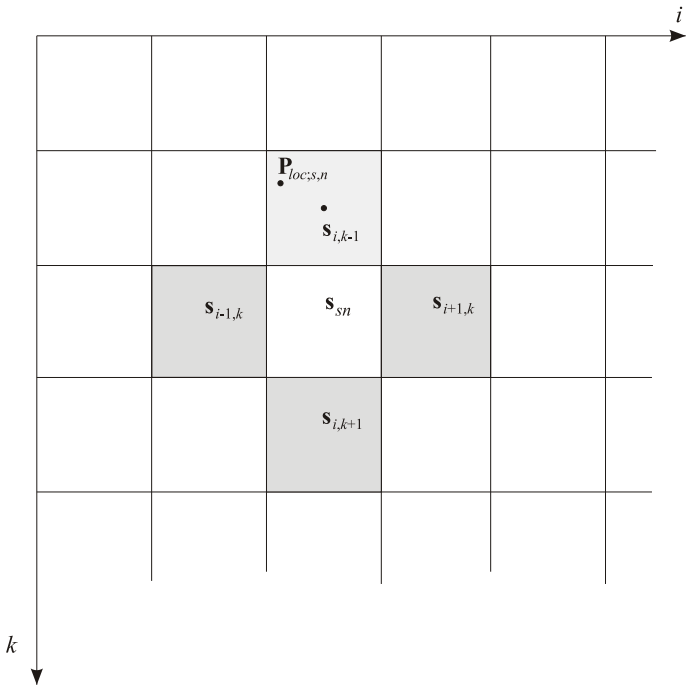
Odmik izberemo kot poljubno sosednjo točko mreže terena glede na pozicijo stare rastline na terenu in na pripadajočem polju kjerkoli posadimo novo rastlino. Sosednje štiri točke za možne nove položaje rasti v bližini staršev so prikazane na sliki C.19.

Algoritem 20 Izboljšan algoritem za dodajanje novih rastlin.

Vhod: N_s - število rastlinskih vrst; $\Delta N_{g,s}$ - število novih rastlin pri tej generaciji pri rastlinski vrsti s; p_s - verjetnost rasti nove rastline blizu svoje vrste za vsako vrsto posebej

Izhod: nove rastline $P_{s,n}$ so dodane rastlinskim vrstam in na teren

for $s = 0$ **to** $N_s - 1$ **do**
  $A_s := 0$;
  for $n = 0$ **to** $\Delta N_{g,s} - 1$ **do**
    if izpolni(p_s) **and** dolžina seznama($\{P_{s,p}\}$)-1≥ 0 **then**
      $I_{s,n}$:= naključni indeks med vključno (0, dolžina seznama($\{P_{s,p}\}$)-1);
    else
      $A_s := A_s + 1$;
    end if
  end for
  uredi seznam $\{I_{s,n}\}$ po vrednostih shranjenih indeksov;
  for $n = 0$ **to** dolžina seznama($\{I_{s,n}\}$) -1 **do**
    $P_{\text{parent};s,n} := I_{s,n}$;
    $\mathbf{s}_{s,n}$:= vrni naključno sosednjo krpo($P_{\text{parent};s,n}$);
    $\mathbf{p}_{s,n}$:= naključna točka na krpi ($\mathbf{s}_{s,n}$);
    if položaj ni v vodi **then**
      $P_{s,n}$:= tvori novo rastlino($\mathbf{p}_{s,n}$); {seme pade na plodna tla}
    end if
  end for
  for $n = 0$ **to** $A_s - 1$ **do**
    $\mathbf{s}_{s,n}$:= vrni naključno krpo(); {dodamo še rastline, ki lahko rastejo daleč stran od staršev}
    $\mathbf{p}_{s,n}$:=naključna točka na krpi ($\mathbf{s}_{s,n}$);
    if položaj ni v vodi **then**
      $P_{s,n}$:= tvori novo rastlino ($\mathbf{p}_{s,n}$);
    end if
  end for
end for

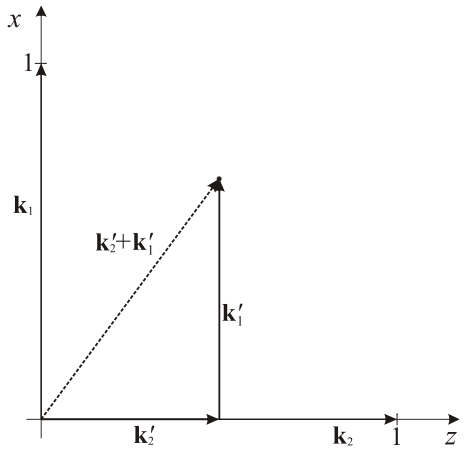


Slika C.19: Primer izbire naključne rasti rastline blizu svojega starša. Krpi $\mathbf{s}_{s,n}$ smo izmed obarvanih možnih krp naključno izbrali sosednjo krpo z manjšim indeksom k. Na tej krpi smo izbrali naključno točko $\mathbf{p}_{s,n}$, ki predstavlja lokacijo nove rastline.

Ob sejanju novih dreves v bližini obstoječih moramo določiti premik v sosednjost. V sosednjo krpo se iz krpe $\mathbf{s}_{s,n}$ premaknemo tako, da naključno izberemo eno izmed štirih sosednjih krp, glede na indeks (i, k), ki ga $\mathbf{s}_{s,n}$ hrani. Tako naključno izberemo enega izmed indeksov $(i-1, k)$, $(i+1, k)$, $(i, k-1)$ in $(i, k+1)$, pri čemer pri robovih terena pazimo, da krpa s takšnim indeksom dejansko obstaja.

Po premiku v sosednjost obstoječe rastline ustvarimo tam novo rastlino. Za n-to krpo $\mathbf{s}_{s,n}$ iz njenega indeksa krpe (i, k) določimo lokacijo nove rastline z naključno izbiro točke na ploskvi krpe $\mathbf{s}_{i,k}$. To storimo tako, da določimo vektorja $\mathbf{k}_1$ in $\mathbf{k}_2$ od izhodiščne točke krpe do skrajnih točk krpe, to je začetnih točk naslednjih krp $(i+1, k)$ in $(i, k+1)$. Z naključnim skaliranjem vektorjev $\mathbf{k}_1$ in $\mathbf{k}_2$ ter seštevanjem dobljenih skaliranih vektorjev dobimo vektor, ki predstavlja iskano lokacijo nove rastline $\mathbf{P}_{loc;s,n}$. Opisan postopek je prikazan na sliki C.20. Pri tem poskrbimo še, da se y-koordinata rastline pravilno prilega površini krpe, kar dosežemo s projiciranjem izbrane točke na to krpo.

Pri tvorbi novih rastlin poleg rastline shranimo njene faktorje, tako da so kasneje lažje dostopni. Inicializiramo starost, velikost, radij, živost in tekmovalno sposobnost rastline na 0. Lokacijo rastline nastavimo iz dobljene lokacije na krpi, prav tako pa

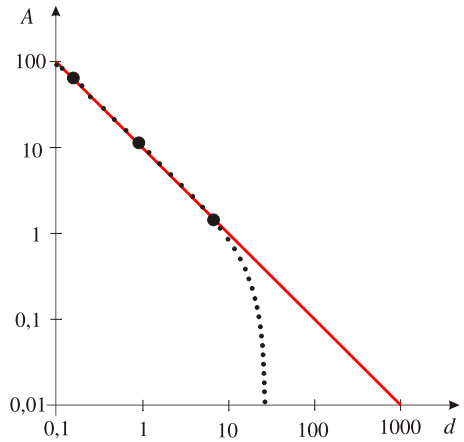


Slika C.20: Določitev naključne točke na krpi.

shranimo indeks krpe, ki ji rastlina pripada. Iz podatkov o terenu odčitamo strmino, višino, vlažnost, osončenost in vetrovnost ter jih shranimo. Za rastlino prav tako določimo največjo možno starost v letih, ki jo lahko doseže.

Tekmovalnost med rastlinami

Ob zgostitvi rastlin na nekem območju upoštevamo, da dominirane rastline lahko odmrejo, kar je posledica samo-redčenja. Ta pravi, da funkcija povprečne teže (debeline rastlin) v odvisnosti od gostote rastlin (števila primerkov) na logaritemski skali pada s strmino $-\frac{3}{2}$ (slika C.21).



Slika C.21: Ricklefsova krivulja samo-redčenja (povprečna površina rastlin v odvisnosti od njihove gostote). Proti nasičenju začnejo gostote manj naraščati, kar pomeni, da nekatere rastline odmirajo.

Da določimo, katera rastlina dominira kateri, vsaki rastlini priredimo tekmovalno sposobnost $a_{s,p}$. To določimo iz produkta živosti rastline $v_{s,p}$, trenutne velikosti rastline

$h_{s,p}$ in **največje možne relativne doživete tekmovalne sposobnosti** $a_{\mathrm{r};p}$:

$$a_{s,p} = v_{s,p} h_{s,p} a_{\mathrm{r};p}. \tag{C.26}$$

Največja možna relativna doživeta tekmovalna sposobnost $a_{\mathrm{r};p}$ predstavlja razmerje med časom največje starosti rastline $t_{\mathrm{f};s}$ za neko rastlinsko vrsto napram času $\max\{t_{\mathrm{f};s}\}$, kar je največja potencialno možna življenjska doba posamezne rastline (npr. $\max\{t_{\mathrm{f};s}\} = 300$ let):

$$a_{\mathrm{r};p} = \frac{t_{\mathrm{f};s}}{\max\limits_{s}\{t_{\mathrm{f};s}\}}. \tag{C.27}$$

Živost rastline $v_{s,p} \in [0,1]$ določimo s seštevanjem ugodnosti pogojev terena na krpi $\mathbf{s}_{i,k}$. Živost je aritmetična sredina vseh pogojev in je prikazana z naslednjo enačbo:

$$v_{s,p} = \frac{k_{\overline{y};s,p} + m_{s,p} + l_{s,p} + w_{s,p} + s_{s,p}}{5}. \tag{C.28}$$

Omenjeni pogoji so nadmorska višina $k_{\overline{y};s,p}$, vlaga $m_{s,p}$, osončenost $l_{s,p}$, vetrovnost $w_{s,p}$ in strmina $s_{s,p}$. Vpliv slednjega izračunamo z upoštevanjem škodljivosti naklona $f_{\mathrm{s};s}$, ki jo podamo za vsako rastlinsko vrsto posebej:

$$s_{s,p} = 1 - (f_{\mathrm{s};s} s_{i,k})^2. \tag{C.29}$$

Indeksi (i,k) označujejo lastnost terena na mestu obravnavane rastline. Ostale štiri pogoje izračunamo glede na odmike od **optimalnih vrednosti lastnosti terena** $\overline{y}_{\mathrm{o};s}$, $m_{\mathrm{o};s}$, $l_{\mathrm{o};s}$ in $w_{\mathrm{o};s}$ po naslednjih enačbah:

$$k_{\overline{y};s,p} = \begin{cases} 1, & |\overline{y}_{i,k} - \overline{y}_{\mathrm{o};s}| \le \Delta\overline{y}_{\mathrm{o};s} \\ 1 - \max\left\{1, |\overline{y}_{i,k} - \overline{y}_{\mathrm{o};s}| f_{\overline{y};s}\right\}, & \text{sicer} \end{cases}, \tag{C.30}$$

$$m_{s,p} = \begin{cases} 1, & |m_{i,k} - m_{\mathrm{o};s}| \le \Delta m_{\mathrm{o};s} \\ 1 - \max\left\{1, |m_{i,k} - m_{\mathrm{o};s}| f_{\mathrm{m};s}\right\}, & \text{sicer} \end{cases}, \tag{C.31}$$

$$l_{s,p} = \begin{cases} 1, & |l_{i,k} - l_{\mathrm{o};s}| \le \Delta l_{\mathrm{o};s} \\ 1 - \max\left\{1, |l_{i,k} - l_{\mathrm{o};s}| f_{\mathrm{l};s}\right\}, & \text{sicer} \end{cases}, \tag{C.32}$$

$$w_{s,p} = \begin{cases} 1, & |w_{i,k} - w_{\mathrm{o};s} \le \Delta w_{\mathrm{o};s} \\ 1 - \max\left\{1, |w_{i,k} - w_{\mathrm{o};s}| f_{\mathrm{w};s}\right\}, & \text{sicer} \end{cases}. \tag{C.33}$$

Dopustne odmike določimo s parametri $\Delta\overline{y}_{\mathrm{o};s}$, $\Delta m_{\mathrm{o};s}$, $\Delta l_{\mathrm{o};s}$ in $\Delta w_{\mathrm{o};s}$. V kolikor je lastnost izven želenih meja, pogoj iz ugodne vrednosti zmanjšamo za produkt tega odmika in **faktorje slabega vpliva** $f_{\overline{y};s}$, $f_{\mathrm{m};s}$, $f_{\mathrm{l};s}$ in $f_{\mathrm{w};s}$.

Odmiranje rastlin

V našem modelu rastlina lahko odmre, če doseže običajno starost odmrtja (življenjska doba) ali je predolgo dominirana od drugih rastlin. Opišimo prvi obravnavan način odmrtja rastline, s pretekom življenjske dobe. Čas naravnega odmrtja rastline $t_{\mathrm{max};s,p}$ naključno določimo pri prvem dodajanju rastline in je med časom starosti $t_{\mathrm{m};s}$ in časom največje starosti $t_{\mathrm{f};s}$:

$$t_{\mathrm{max};s,p} = t_{\mathrm{m};s} + (t_{\mathrm{f};s} - t_{\mathrm{m};s}) f_{\mathrm{random}}, \qquad f_{\mathrm{random}} \in [0, 1]. \tag{C.34}$$

Drugi način odmrtja rastline je dominiranost. Če rastlina A s tekmovalno sposobnostjo a_A raste poleg rastline B, ki ima večjo tekmovalno sposobnost a_B, postane dominirana. V tem primeru raste počasneje ali sploh ne in lahko odmre z verjetnostjo $p_{m;s}$ (nastavljiv parameter simulacije), ki je določena za vsako rastlinsko vrsto posebej in modelira odpornost na rast v sobivanju.

Kot pomoč za nadaljnjo razlago določimo število krp N_{opt}. To določa, za koliko krp sta največ lahko oddaljeni poljubni rastlini, da se njuna ekološka kroga še lahko prekrivata. Ugotovimo lahko, da je to natanko toliko, čez koliko krp se lahko razteza premer ekološke sosednosti največje rastline. Ta pa je enak največjemu možnemu premeru R_s ekološkega kroga za rastlinsko vrsto. Če s K označimo dimenzije krpe, podaja izračun N_{opt} naslednja enačba:

$$N_{\mathrm{opt}} = \left\lceil \frac{2 \max\limits_{s} \{R_s\}}{K} \right\rceil. \tag{C.35}$$

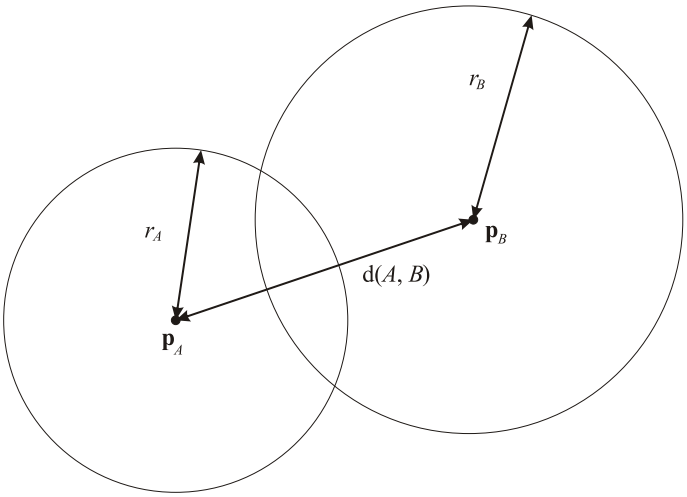
Potencialno močnejše rastline glede na trenutno rastlino na krpi (i, k) poiščemo tako, da preverimo tekmovalne sposobnosti rastlin v okolici te krpe. Če bi preiskovali celoten teren, bi simulacija potekala prepočasi, zato pregledamo le potencialno število sosednjih krp N_{opt}. Da lahko obiščemo le N_{opt} krp, si ponovno pomagamo z referenčnim seznamom lokacij rastlin po krpah, ki smo ga predstavili že pri dodajanju rastlin v bližini starševskih rastlin. V seznamu imamo za vsako krpo shranjene rastline, ki na njej rastejo. Vse rastline na neki krpi primerjamo z obravnavano rastlino in preverimo, ali se ekološki krogi prekrivajo ter ali ima obravnavana rastlina manjšo tekmovalno sposobnost. Če sta pogoja izpolnjena, prenehamo z iskanjem in obravnavano rastlino določimo kot dominirano.

Geometrijsko razdaljo $d(A, B)$ med rastlinama A in B na lokacijah $\mathbf{p}_A$ in $\mathbf{p}_B$ določimo z:

$$d(A, B) = \|\mathbf{p}_A - \mathbf{p}_B\|. \tag{C.36}$$

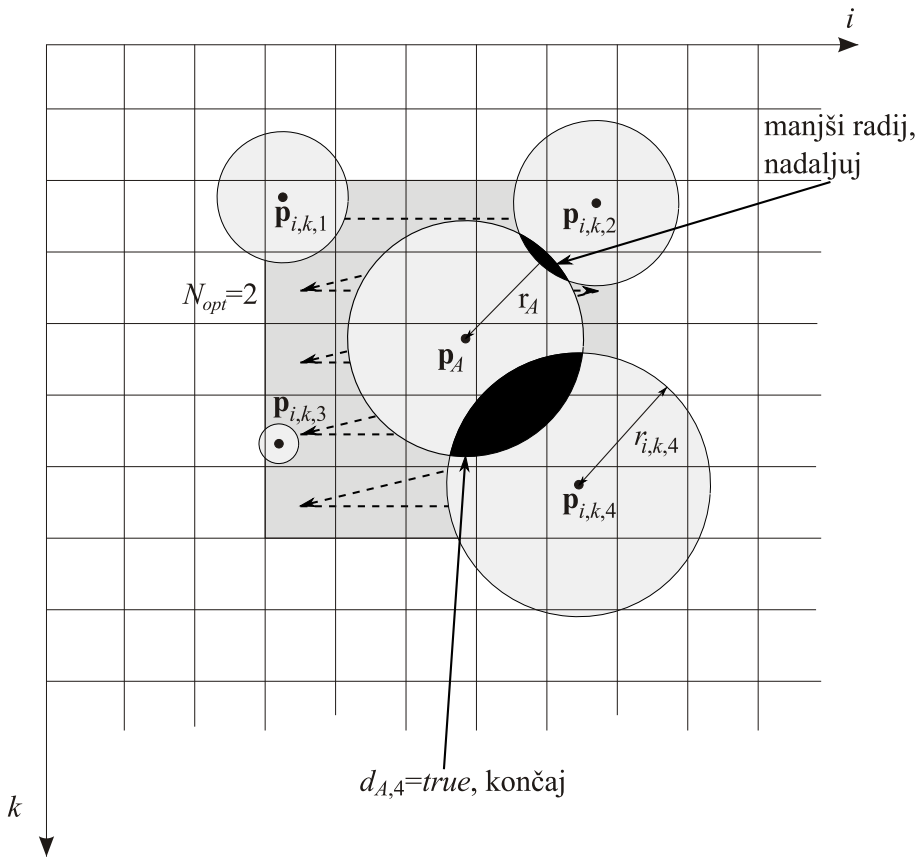
Prekrivanje ekoloških krogov dveh rastlin $F_{A,B}$ (glej sliko C.22) določimo iz geometrijske razdalje med rastlinama. Če je razdalja med rastlinama manjša od vsote radijev r_A in r_B, se ekološka kroga sekata. Dokler ne preverimo vseh bližnjih rastlin in dokler katera od njih ne dominira rastline A, izračunavamo prekrivanje z rastlinami B:

$$F_{A,B} = \begin{cases} da, & d(A,B) < r_A + r_B \\ ne, & \text{sicer} \end{cases}. \tag{C.37}$$



Slika C.22: Dominiranost izračunamo iz prekrivanja ekoloških krogov.

Rastline za dominiranost iščemo tako na polju krp vključno med krpama $(i - N_{\text{opt}}, k - N_{\text{opt}})$ in $(i + N_{\text{opt}}, k + N_{\text{opt}})$ (slika C.23). V primeru, da se ekološka kroga rastlin prekrijeta in ima sosednja rastlina boljšo tekmovalno sposobnost, končamo s preiskovanjem in vrnemo *da*, sicer *ne*. Primerjati moramo tekmovalne sposobnosti omenjenih rastlinih, ne zgolj velikosti radijev, saj na radije poleg velikosti rastlin $h_{s,p}$ vpliva R_s (enačba C.41), na tekmovalo sposobnost rastline pa poleg velikosti rastlin $h_{s,p}$ vpliva še živost rastline $v_{s,p}$ in največja možna doživeta tekmovalna sposobnost rastline $a_{\text{r;}p}$ (enačba C.26). Zaporedje pregledovanja krp označuje črtkana črta. Slika C.23 prikazuje primer s štirimi rastlinami, ki jih lahko označimo po vrsti z indeksi 1, 2, 3 in 4. Položaj rastline $\mathbf{p}_A$ po vrsti primerjamo s položaji $\mathbf{p}_{i,k,1}$, $\mathbf{p}_{i,k,2}$, $\mathbf{p}_{i,k,3}$ in $\mathbf{p}_{i,k,4}$. Ker se prvi krog ekoloških sosednosti rastlin ne seka s krogom ekološke sosednosti osnovne rastline, nadaljujemo. Čeprav se v drugem primeru zgodi presek, pa je tokrat primerjana rastlina prešibka, zato ponovno nadaljujemo. V primeru tretje sosednje rastline prav tako nadaljujemo iskanje. Ustavimo se pri četrti rastlini, ki dominira osnovno rastlino, saj za njuni tekmovalni sposobnosti velja $a_A < a_{i,k,4}$ in njuna ekološka kroga se prekrivata ($|\mathbf{p}_A - \mathbf{p}_{i,k,4}| < r_A + r_{i,k,4}$). Postopek je opisan z algoritmom 21.



Slika C.23: Iskanje dominiranosti rastlin v omejenem območju okoli rastline A, $N_{\text{opt}} = 2$.

Algoritem 21 Iskanje dominiranosti.

Vhod: $\mathbf{p}_A$ - položaj rastline A; a_A - tekmovalna sposobnost rastline A; r_A - polmer rastline A; i, k - indeksa krpe obravnavane rastline A; N_{opt} - velikost okolice za iskanje po sosednjih krpah seznam $\mathbf{p}_{i,k,n}$ - lokacije rastlin n, ki so blizu krpe $\mathbf{s}_{i,k}$; seznam $a_{i,k,n}$ - tekmovalne sposobnosti rastlin n, ki so blizu krpe $\mathbf{s}_{i,k}$; seznam $r_{i,k,n}$ - radiji rastlin n, ki so blizu krpe $\mathbf{s}_{i,k}$;

Izhod: za obravnavano rastlino določimo zastavico $D_{A,B}$ - ali je dominirana od katerekoli bližnje rastline

```
D_{A,B} := ne;
for z = k − N_opt to k + N_opt do
  for x = i − N_opt to i + N_opt do
    for n = 0 to dolžina seznama({p_{i,k}})-1 do
      if |p_A − p_{i,k,n}| < r_A + r_{i,k,n} and a_A < a_{i,k,n} then
        D_{A,B} := da;
        return;
      end if
    end for
  end for
end for
```

Reprodukcija rastlin

Rastline vrste s se po terenu reproducirajo glede na število svojih semen na terenu s_s in faktor razploda te vrste g_s (nastavljiv parameter simulacije, ponavadi celo število, npr. od 1 do 5). Število novih rastlin ΔN_s vrste s, ki vzkalijo v trenutnem koraku simulacije, izračunamo po naslednji enačbi:

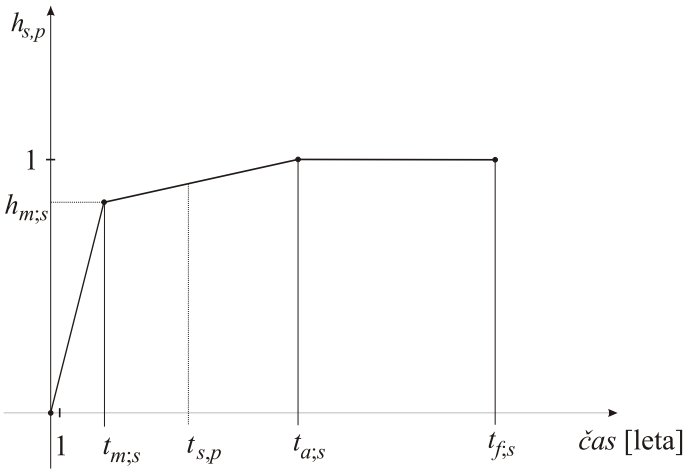
$$\Delta N_s = s_s g_s, \qquad g_s \in [0, 100]. \tag{C.38}$$

Zbiranje semen določimo z vsoto tekmovalnih sposobnosti rastlin, ki so že zrele in tvorijo semena. Tako večja drevesa dajejo več semen kot manjša. V prvem koraku postavimo $\Delta N_s = 1$. Iz enačbe C.38 sledi, da so kumulacije semen za vse rastlinske vrste enake $\frac{1}{g_s}$ in zato nastane po ena rastlina na rastlinsko vrsto, kasneje pa semena ustrezno dodajamo. Povezavo med tekmovalnimi sposobnostmi rastlin $a_{s,p}$ (enačba C.26, stran 123) in kumulacijo semen vseh $N_{g,s}$ rastlin iste vrste podaja enačba C.39. Ker je prava tekmovalna sposobnost rastline enaka $a_{s,p}$, jo v izračunu označimo s $a'_{s,p}$, kar pomeni, da postavimo tekmovalno sposobnost na 0, dokler rastlina še ni zrela.

$$s_s = \sum_{p=0}^{N_{g,s}} a'_{s,p}, \qquad a'_{s,p} = \begin{cases} a_{s,p}, & t_{s,p} \geq t_{\mathrm{m};s} \\ 0, & \text{sicer} \end{cases}. \tag{C.39}$$

Simulacijski koraki predstavljajo leta. Starost p-te rastline iz rastlinske vrste s označimo s $t_{s,p}$. Mlade rastline rastejo hitreje, v zrelosti rastejo počasneje, v starosti pa več ne rastejo. Če so rastline dominirane (opisano kasneje), rastejo zelo počasi ali sploh ne (v modelu smo upoštevali le slednje). Hitrost rasti rastline spreminjamo v odvisnosti od prej omenjenih treh obdobij rastline in koeficienta ugodnosti pogojev v_p (opisan kasneje). V mladosti rastlina zgolj hitro raste. Zrelost pomeni, da rastlina prvič odvrže svoja semena in iz teh lahko nastanejo nove rastline. Starost poimenujemo obdobje, ko rastlina več ne raste. Starost traja največ do leta $t_{\mathrm{f};s}$, ko vsaka rastlina te vrste zagotovo odmre. Meji med opisanimi obdobji predstavljata leto zrelosti (ang. *matured*) $t_{\mathrm{m};s}$ in leto starosti (ang. *aged*) $t_{\mathrm{a};s}$ posamezne rastlinske vrste. Odvisnost med značilnimi starostmi rastline in hitrostjo rasti je prikazana na sliki C.24. Za vsako vrsto prav tako podamo višino $h_{\mathrm{m};s}$, ki jo rastlina doseže v letu zrelosti. S podanimi parametri lahko izračunamo prirast k velikosti rastline $\Delta h_{s,p}$ v enem simulacijskem ciklu po enačbi:

$$\Delta h_{s,p}[\mathrm{m/leto}] = \begin{cases} v_p \frac{h_{\mathrm{m};s}}{t_{\mathrm{m};s}}, & 0 \leq t_{s,p} \leq t_{\mathrm{m};s} \\ v_p \frac{1 - h_{\mathrm{m};s}}{t_{\mathrm{a};s} - t_{\mathrm{m};s}}, & t_{\mathrm{m};s} \leq t_{s,p} \leq t_{\mathrm{a};s} \\ 0, & \text{sicer} \end{cases}. \tag{C.40}$$

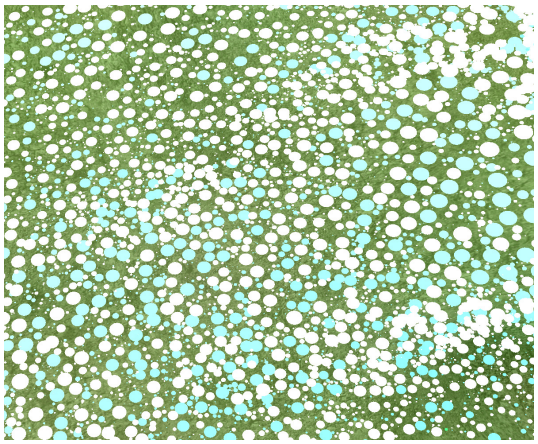


Slika C.24: Hitrost rasti skozi staranje. Do zrelosti rastlina raste hitreje, v času zrelosti raste počasneje, v starosti pa rastlina več ne raste. S $h_{\mathrm{m};s}$ podamo velikost rastline ob času zrelosti.

Velikost ekološkega radija $r_{s,p}$ rastline izračunamo iz višine rastline $h_{s,p} \in [0,1]$ (normaliziranost izhaja iz enačbe C.30, stran 123) s pomočjo parametra R_s za razmerje med starostjo v letih in velikostjo radija v metrih (uporabniško podan za vsako rastlinsko vrsto posebej):

$$r_{s,p} = h_{s,p} R_s, \qquad R_s \in (0, 10). \tag{C.41}$$

Slika C.25 prikazuje ekološke sosednosti rastlin, ki so odvisne od velikosti rastlin.



Slika C.25: Krogi predstavljajo radije rastlin: večje rastline imajo večje radije. Kot vidimo, se rastline dokaj malo prekrivajo. Na začetku simulacije je bilo veliko manjših rastlin, nato pa se je število rastlin močno zmanjšalo, saj so začele prevladovati večje rastline.

Potek simulacije evolucijskega modela

Simulacijo začnemo z neporaslim terenom, na katerega vsako leto dodamo določeno število rastlin posamezne vrste. Pri odvijanju simulacije opazimo želeno redčenje, ki je eden od osnovnih naravnih principov ekologije rastlin. Na danem območju se tako najprej razvije več manjših rastlin, kasneje pa prevladajo večje, ki jih je ustrezno manj.

Opisani pogoji pokrajine vplivajo na rastline tako, da rastline, ki ne tolerirajo določenega življenjskega pogoja na svojem območju rasti, počasneje rastejo in kasneje skozi simulacijske korake odmrejo s podano verjetnostjo na simulacijski korak. Med rastjo rastline pridobivajo tekmovalno sposobnost.

V vsakem koraku simulacije izvedemo dodajanje, rast in odstranjevanje rastlin, kot je predstavljeno z algoritmom 22. Naštete rutine se obnašajo stohastično; pri tem uporabljajo naključni generator števil *SUPER-DUPER* z obliko $LCG(2^{32}, 69069, 0, 1)$ [149]. Uveljavitev verjetnostne trditve smo izvedli, če se je normalizirano generirano naključno število uvrstilo pod določen prag a:

$$izpolni(a) = \begin{cases} da, & F(RNi) < a \\ ne, & \text{sicer} \end{cases} . \tag{C.42}$$

Algoritem 22 Nadzorna rutina simulacije.

Vhod: v - seznam rastlinskih vrst; r - seznam rastlin za rastlinske vrste; f - seznam matrik za vrednosti faktorjev na površju;
Izhod: izvaja koračno simulacijo razvoja ekosistema

```
loop
    dodaj nove rastline rastlinskim vrstam(v, r);
    zrasti vse rastline (r, f);
    odstrani odmrle rastline(r);
end loop
```

C.2.3 Animacija drevesnega ekosistema na pokrajini

Pri vizualizatorju dreves na pokrajino uporabljamo rezultate tekočega koraka simulacije in animiramo rast dreves. Za prikaz uporabimo tehnologijo OpenGL. Pred upodabljanjem upoštevamo še obrezovanje scenskih objektov na vidni volumen in poenostavljanje drevesne geometrije.

Tako v sceni prikažemo le trenutno vidne rastline [179, 86], ki se poenostavijo glede na oddaljenost od gledišča. Poenostavitev dreves izvedemo tako, da izrišemo manj

vej in listov za poenostavljeno drevo. Zaenkrat poenostavljamo samo drevesa, saj izrisovanje terena ni časovno potratno napram času izrisovanja dreves.

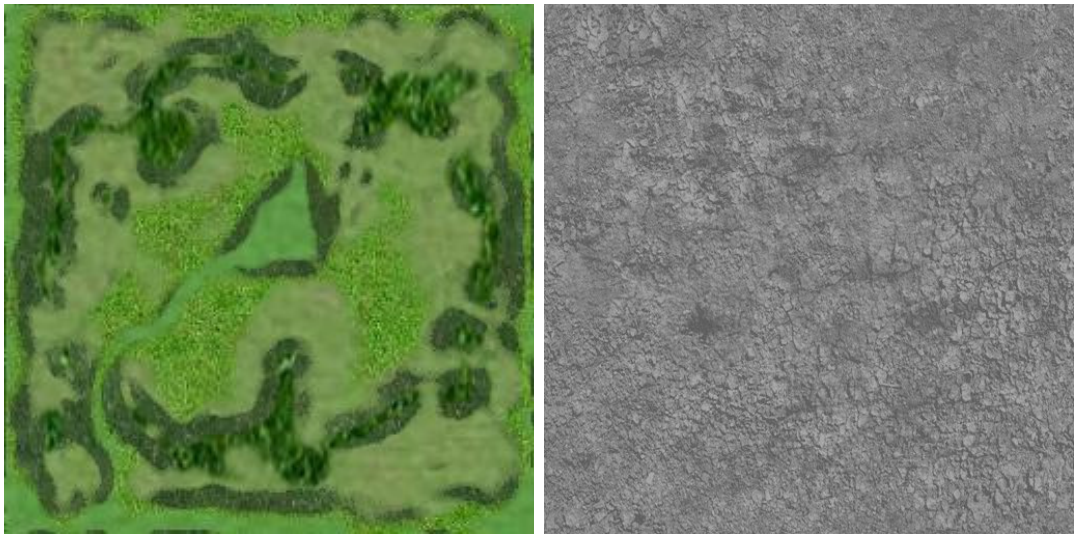
Površje pokrajine upodobimo s trikotniki, ki jih dobimo po triangulaciji množice točk terena. Ker so te točke v ravnini xz razporejene ekvidistančno, jih zapišemo v dvodimenzionalno polje točk, ki ga lahko naslavljamo s celoštevilskimi indeksi. Z naslavljanjem točk je možno hitro sestaviti štirikotnike, ki pokrijejo celoten teren, štirikotnike pa delimo na polovico in dobimo potrebne trikotnike.

Prevedena polja oglišč in multitekstu riranje ploskev v OpenGL

Upodobitev terena smo opravili po specifikaciji OpenGL verzije 1.2 [240, 242], ki omogoča posredovanje **prevedenih polj oglišč** (*compiled vertex arrays*). To pomeni, da na odjemalcu sestavimo vse potrebne točke in povezave med njimi za gradnjo trikotnikov v obliki polja. Polje pošljemo na strežnik OpenGL in ga zaklenemo, tako da ga strežnik shrani v pomnilnik grafične kartice. Uporaba te tehnike pohitri izris mreže terena za več kot 25-krat napram implicitnemu podajanju vsake točke posebej v vsakem novem okvirju animacije [65].

Za uporabo ukazov iz specifikacije OpenGL 1.2 moramo v programsko kodo vključiti ustrezne glave za delo z razširitvami OpenGL in prevedeno objektno kodo povezati s knjižnico za OpenGL. Razširitev, ki nam omogoča delo s prevedenimi polji oglišč, se imenuje `GL_EXT_compiled_vertex_array`. Ta nam omogoča, da s funkcijo grafične knjižnice **DrawElements** izrišemo seznam gradnikov, kot so na primer trikotniški trakovi.

Teksturiranje objektov je opravljeno z nalaganjem rastrskih slik, ki jih posredujemo stroju OpenGL. Ta bitne podatke shrani in vrne ime teksture, s katero nadalje operiramo. Z razširitvijo OpenGL za več tekstur na eni površini dosežemo čistejše slike. Razširitev se imenuje `GL_ARB_multitexture` in omogoča izbiro aktivne med več teksturami s funkcijo **glClientActiveTextureARB**. Tej podamo parameter `GL_TEXTURE<texid>_ARB`, pri čemer `texid` predstavlja indeks izbirane teksture: 0, 1 ali več. Teksture se med seboj kombinirajo. Primer teksture čez celi teren prikazuje slika C.26, leva tekstura. Ker je slika manjše ločljivosti, preko nje nalepimo teksturo za dodajanje podrobnosti. Prelepljena tekstura za podrobnosti na terenu pa je vidna na desni teksturi na sliki C.26 in s tem renderiran model naredimo bolj informacijsko polno in s tem videti bolj podroben.



Slika C.26: Tekstura terena za čez celi teren (levo) in tekstura za dodajanje podrobnosti, visokih frekvenc (desno).

Navigacija

Aplikacija omogoča dva načina pogleda. Prvi prikazuje eno samo drevo v polni podrobnosti izrisa in je posebej primeren za modeliranje drevesa. Med modeli dreves lahko med modeliranjem preklapljamo in tako nastavimo parametre proceduralnih modelov za izračun geometrije vsem vrstam rastlin v ekosistemu. V drugem načinu pogleda na ekosistem je prikazanih veliko dreves na terenu, s prilagojeno podrobnostjo izrisa. Namenjen je opazovanju poteka simulacije.

V načinu pogleda na eno samo drevo gledišče ostaja fiksno, premikamo pa drevo. Viden del drevesa določimo s sukanjem drevesa okoli glavne osi debla in okoli osi, pravokotne na glavno os drevesa. Drevo lahko pomikamo v treh osnih smereh.

V načinu pogleda na ekosistem premikamo gledišče. Parametre proceduralnih modelov dreves lahko sicer nastavljamo tudi v tem načinu pogleda, vendar ga pogosteje uporabimo za oddaljen pogled na celoten ekosistem. Pogled nastavimo s šestimi prostostnimi stopnjami, s katerimi lahko nastavimo poljuben položaj gledišča in smer gledanja. Postavljanje pogleda v načinu ekosistema poteka s klicem funkcije **gluLookAt** iz pomožne grafične knjižnice, ki mu podamo točko gledanja, referenčno točko in vektor v smeri navzgor.

Upodobljene slike simulacij drevesnih ekosistemov

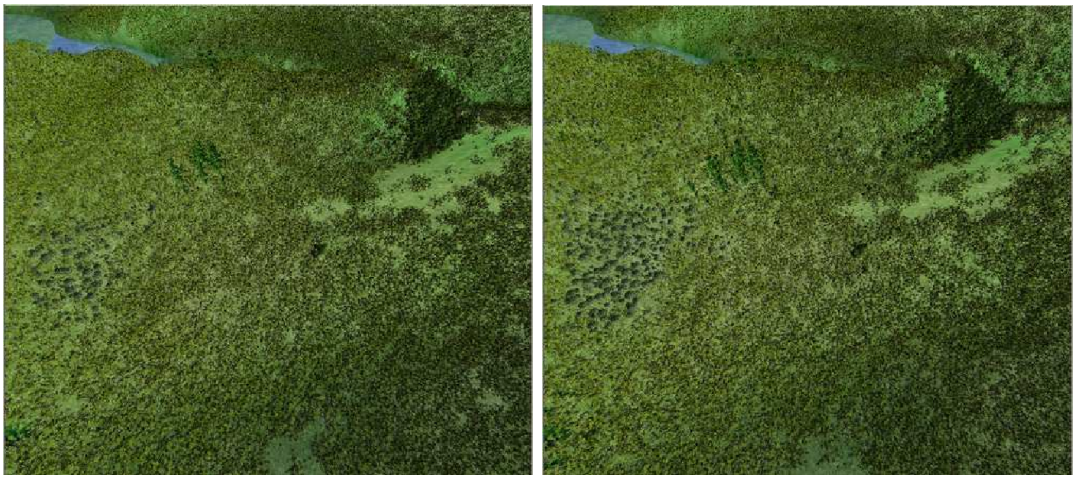
S simulacijo tvorimo porazdelitev rastlin, ki jih lahko upodobimo kot žične modele ali s polnimi teksturami. Namesto rastlin lahko upodobimo tudi samo kroge sosednosti, kar

Slika C.27: Levo) začetek simulacije: človek preneha s košnjo trave, pričnejo se pojavljati grmovnice; desno) čez 20 let: grmovnice se hitro razširijo po terenu.

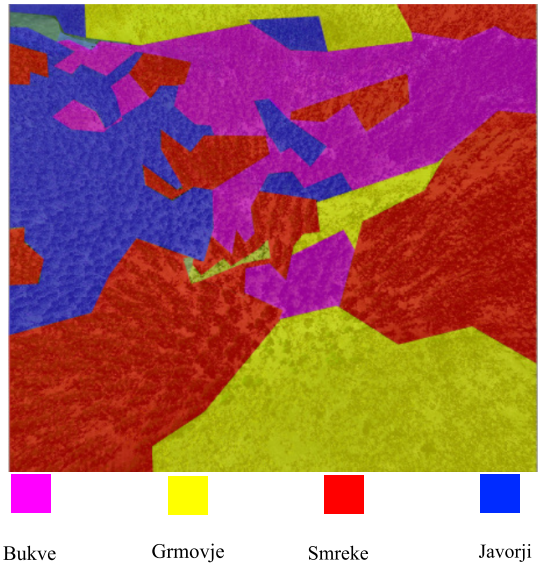
je primerno za shematski prikaz. V nadaljevanju so predstavljene upodobljene slike, ki so dobljene z enim od tekov simulacije. V simulacijo smo vključili štiri vrste dreves: bukve, javorji, smreke in grmovnice.

Simulacijo pričnemo s praznim terenom brez dreves, na katerega se ta naselijo, kot bi človek pokrajino pustil neobdelano (slika C.27). Ker se grmovnice po terenu najhitreje razrastejo, je teh po prvih nekaj letih največ (slika C.27). Te so dokaj šibke v boju za prostor proti večjim drevesnim vrstam, zato se začno umikati in teren začno poseljevati večji listavci (slika C.28). Kasneje se iz desne strani vse bolj začno širiti tudi smreke (slika C.28). Po približno 180 letih simulacije na terenu rastejo vsa uporabljena drevesa v simulaciji. Vsaka vrsta od njih si je za življenjski prostor izborila tisti del, ki je zanjo najugodnejši in kjer lahko izpodrine ostale vrste. Na sliki C.29 lahko vidimo tudi želeno rast dreves v gručah, še vedno pa je med drevesi ene vrste nekaj dreves iz kakšne druge drevesne vrste, kar sliko naredi bolj verodostojno.

Nadaljnje raziskave glede animacije ekosistemov bi bilo možno opraviti pri modeliranju, simulaciji kot tudi pri upodabljanju. Drevesa bi lahko generirali s kakšnim drugim modelom. Modeliranje ekosistema bi lahko opravili z drugačnimi modeli za izračun lastnosti terena. Model bi prav tako lahko dodatno preverili z realnimi podatki iz kakšne slovenske pokrajine. Z razširitvijo modela bi lahko prikazali vpliv onesnaževanja okolja na rast dreves. Simulacijo bi lahko za še hitrejše izvajanje porazdelili z vmesnikom MPI. Upodabljanje bi lahko med več OpenGL strežnikov porazdelili tako, da bi uporabili sistem Chromium, vsak strežnik pa bi izračunal geometrijo proceduralnih modelov dreves in izrisal samo tista, ki so vidna v pripadajočem delu vidnega polja.



Slika C.28: Levo) simulacija po 60 letih: rasti začnejo tudi druge drevesne vrste, ki izpodrivajo grmovnice, teren je zaraščen; desno) simulacija po 80 letih: vse bolj prodirajo tudi smreke.



Slika C.29: Simulacija po 180 letih: bukve, javorji, smreke in grmovnice se porazdelijo glede na lastnosti terena in medsebojne vplive.

D. Interaktivno gojenje ekosistemov

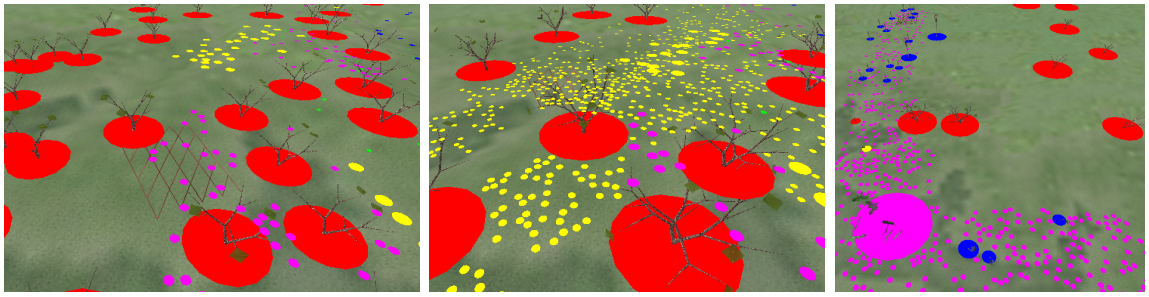
History will be kind to me, for I intend to write it. (Winston Churchill)

Nad ekosistemom smo izvedli tudi interaktivno evolucijo parametrov. Pri tem smo uporabili literaturo iz področja evolucijskih algoritmov [255] in umetnega življenja [273]. Uvedbo gojenja ekosistemov v L-sisteme najdemo v [234], primer evolucije in mutacije prikazuje slika D.1.

V sistemu EcoMod je omogočeno upodabljanje velike količine dreves, ki ga dobimo iz simulacija ekosistema kot paralelne nit izvajanja programa. Sistem EcoMod omogoča tudi modeliranje izgleda drevesnih vrst, kot smo že predstavili. V simulaciji evolucije ekosistema je število skalarnih parametrov enako $D_s = 26$ za vsako vrsto in $D_e = 7$ za vsako krpo. Simulacija evolucije vsebuje do $NP = 500.000$ primerkov dreves na npr. 10.000 krpah terena, kar daje več kot 1e+7 spremenljivk, ki vplivajo na to simulacijo.

Za interaktivno gojenje ekosistema smo kot svoj vidik pripravili interaktivni vmesnik do obstoječega evolucijskega modela v sistetmu EcoMod in dodali nameščanje vpliva katastrof, ki lahko spreminjajo lastnosti okolja. Za interaktivno gojenje ekosistemov smo sedaj dodatno vključili interakcijo preko OpenGL funkcije `gluUnProject()`, ki daje 3D položaj kazalca na sceni. Uvedli smo naslednje evolucijske operatorje: selekcija (2x), mutacija (2x), križanje (1x) in dinamika okolja (2x). Uporabnost tega vidimo v hitri simulaciji scenarijev pogozdovanja ali vrtnarjenja ter animaciji in zabavni industriji.

Slika D.1: Gojenje ekosistemov po postopku [234]: pričetek evolucije (levo) in mutacija z radiacijo (desno); vir: [234].



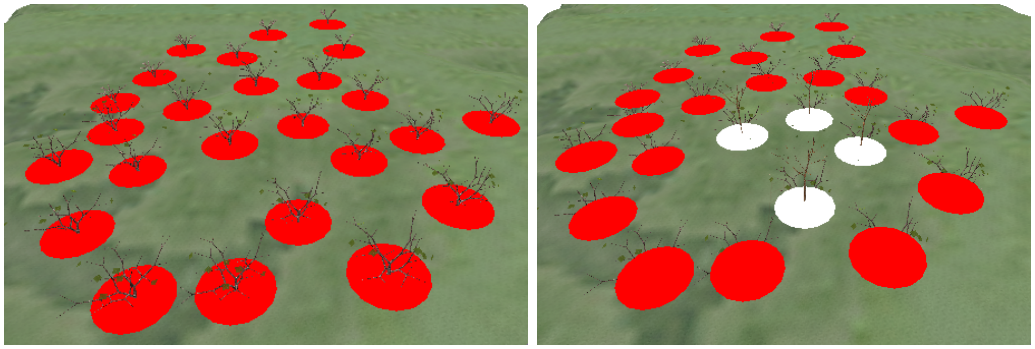
Slika D.2: Operator selekcije: dodajanje.

Na simulacijo evolucijskega procesa v ekosistemu bi lahko gledali kot naslednje evolucijske operatorje, ki nanjo vplivajo: **variacija (mutacija)** – drevo z rastjo (skozi simulacijski čas) spreminja svoje lastnosti, **ocena ustreznosti** – primerki z rastjo pridobivajo moč (*fitness*), ki jim daje večjo verjetnost za preživetje v boju za naravne vire in **koevolucija** – primerki medsebojno tekmujejo za prevlado na določenem mestu in ocena za selekcijo je relativna (*relative fitness*). Tako si za cilj v simulaciji lahko zadamo iskanje dinamičnega ekvilibrija porazdelitve rastlin v ekosistemu, ki je večkriterijski optimizacijski problem. Za določitev izida simulacije lahko izberemo tudi tehniko umetnega življenja. V simulacijo smo tako dodali možnost interakcije z operatorji, kot so **selekcija** – dodajanje, brisanje, **mutacija** – nova vrsta, naključna variacija, **križanje** – izmenjava genomskih informacij in **dinamika okolja** – variacija posameznih pogojev, ali variacija terena za ponovno oceno pogojev.

Da bi modelirali selekcijski operator, smo najprej uvedli *dodajanje*. S selekcijskim operatorjem za dodajanje rastlin na teren dodamo rastline na naključnem mestu v bližnji okolici vpliva selekcijskega operatorja. Primer njegove uporabe vidimo na sliki D.2, kjer smo dodajali večje število določene vrste rastlin v različne scenarije.

Za selekcijski operator smo uvedli še brisanje rastlin, ki označene rastline odstrani iz simulacije. Selekcijski operator za brisanje rastlin iz terena odstrani vse rastline, ki se

Slika D.3: Operator selekcije: pred in po brisanju.

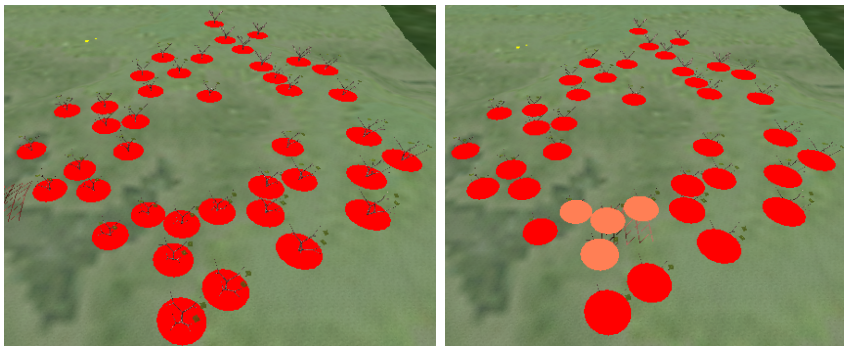


Slika D.4: Operator mutacije: pred in po mutaciji.

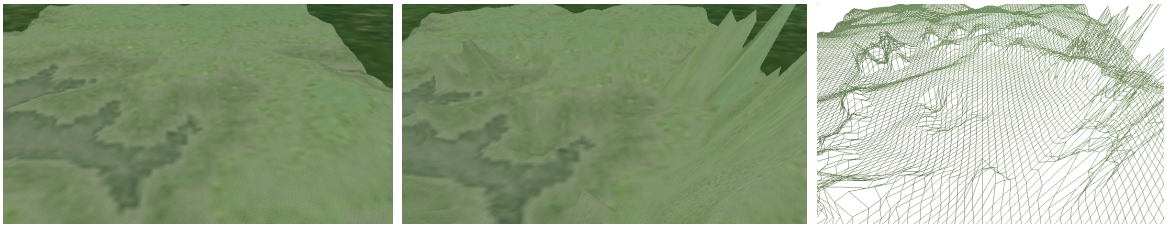
nahajajo v bližnji okolici vpliva selekcijskega operatorja. Pri tem operatorju koristimo podatkovno strukturo za shranjevanje rastlin iz slike C.23, ki iz enakomerne delitve terenske ravnine hrani referenčne sezname primerkov rastlin $p_{i,k,n}$. Primer uporabe vidimo na sliki D.3, kjer iz terena odstranimo drevo desno spodaj.

Z mutacijo spremenimo lastnosti vrste rastlin. Lastnosti vrste rastlin so vsebovane v genomu oz. semenu, ki sestoji iz 26 parametrov (ugodnosti rasti in druge karakteristike). Pri mutaciji vse parametre spremenimo za uniformno naključen delež $F_{\text{mut}} = 0{,}05$.

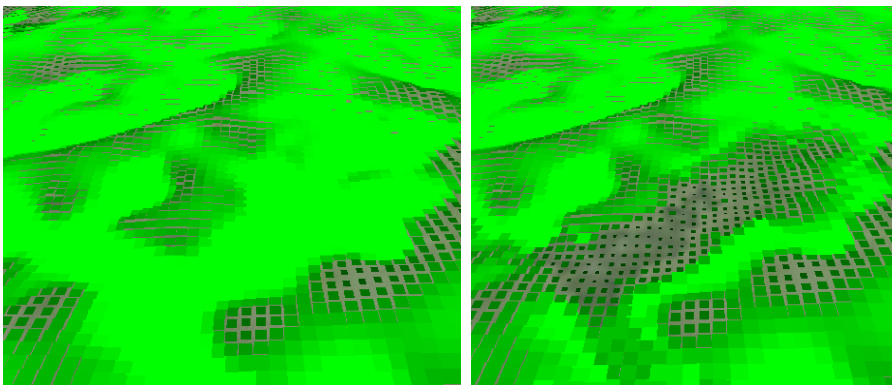
Ob večji mutaciji gena zaradi različnih vzrokov uvedemo novo vrsto. Za ta namen smo uvedli nov operator: nova vrsta. Mutacija genoma, ki povzroči nastanek nove, ločene vrste v okolici vpliva operatorja, vse rastline poenotimo v isto, novo vrsto, kar povzroči ločitev vrste (t.j. ta skupina dobi svojo gensko vrsto). Kot je znano iz biologije, nastanek vrste ni pogojen z genomom, temveč ločitvijo rekombinacijskih populacij. Tako z izmenjavo genomskih informacij pri križanju vse primerke v okolici združimo v novo vrsto in nov genom sestavimo proporcionalno glede na zastopanost vključenih rastlin v vzorcu. Primer interaktivnega operatorja za mutacijo v novo vrsto vidimo na sliki D.5, kjer smo drevesa na sredini označili kot novo vrsto.



Slika D.5: Operator križanja: pred in po križanju.

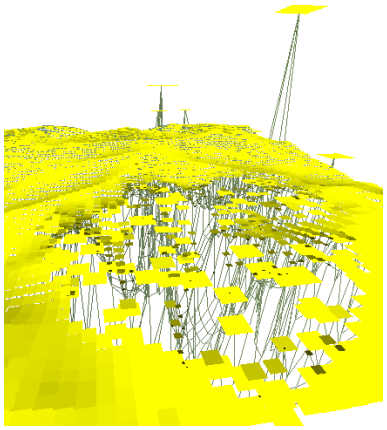


Slika D.6: Operator nad terenom: pred (levo) in po spremembi (v sredini in desno, polnjen in žični model).



Slika D.7: Operator nad parametri okolja: pred in po spremembi vlažnosti.

Ker želimo nadzorovati tudi površje, na katerem drevesa v ekosistemu rastejo, smo omogočili tudi določeno mero dinamike okolja. Interaktivno oblikovanje terena vpliva na razdalje med drevesi, lahko pa glede na spremenjen teren s simulacijami ponovno ocenimo tudi življenjske pogoje. Primer spreminjanja terena vidimo na sliki D.6, kjer smo speminjali višine terena na desni strani. Variirali smo tudi življenjske pogoje na terenu (vlaga, sonce, veter), kot je vidno na slikah D.7 in D.8 za vlago in veter.



Slika D.8: Operator nad parametri okolja: po spremembi vetrovnosti.

Literatura

[1] H. A. Abbass. The self-adaptive Pareto differential evolution algorithm. V *Proceedings of the 2002 Congress on Evolutionary Computation, 2002*, številka 1, strani 831–836, 2002.

[2] H. A. Abbass, R. Sarker in C. Newton. PDE: A Pareto-frontier Differential Evolution Approach for Multi-objective Optimization Problems. V *Proceedings of the Congress on Evolutionary Computation 2001*, številka 2, strani 971–978, Piscataway, New Jersey, 2001. IEEE Service Center.

[3] B. Alatas, E. Akin in A. Karci. MODENAR: Multi-objective differential evolution algorithm for mining numeric association rules. *Applied Soft Computing*, 8(1):646–656, 2008.

[4] M. M. Ali. Differential evolution with preferential crossover. *European Journal of Operational Research*, 127(3):1137–1147, 2007.

[5] M. Aono in T. Kunii. Botanical tree image generation. *IEEE Computer Graphics and Applications*, 4(5):10–34, 1984.

[6] B. V. Babu in M. M. L. Jehan. Differential Evolution for Multi-Objective Optimization. V *Proceedings of the 2003 Congress on Evolutionary Computation (CEC 2003)*, številka 4, strani 2696–2703, Canberra, Australia, 2003. IEEE Press.

[7] B. V. Babu, J. H. S. Mubeen in P. G. Chakole. Multiobjective Optimization Using Differential Evolution. *TechGenesis – The Journal of Information Technology*, 2(2):4–12, 2005.

[8] T. Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, New York, 1996.

[9] R. Balling, B. Theodore in R. Wilkins. Engineering Hyperstructures for Cities of the Future. V *Structures Congress 2010*, strani 3198–3206. ASCE, 2010.

[10] L. Barone, L. While in P. Hingston. Designing Crushers with a Multi-Objective Evolutionary Algorithm. V *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002)*, uredniki: W. B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke in N. Jonoska, strani 995–1002, San Francisco, California, 2002. Morgan Kaufmann Publishers.

[11] N. Barricelli. Numerical testing of evolution theories. *Acta Biotheoretica*, 16(1):69–98, 1962.

[12] D. Beaumont in S. Stepney. Grammatical Evolution of L-systems. V *The 2009 IEEE Congress on Evolutionary Computation CEC 2009*, strani 2446–2453. IEEE Press, 2009.

[13] G. Beni in J. Wang. Swarm intelligence in cellular robotic systems. *Robots and Biological Systems: Towards a New Bionics?*, strani 703–712, 1993.

[14] H.-G. Beyer in H.-P. Schwefel. Evolution strategies — A comprehensive introduction. *Natural Computing*, 1:3–52, 2002.

[15] J. Bloomenthal. Modeling the mighty maple. V *SIGGRAPH '85 Conference Proceedings, San Francisco, CA, 22–26 July 1985*, urednik: B. A. Barsky, strani 305–311, 1985.

[16] P. Boggs in J. Tolle. Sequential quadratic programming for large-scale nonlinear optimization. *Journal of Computational and Applied Mathematics*, 124(1-2):123–137, 2000.

[17] B. Bošković, J. Brest, A. Zamuda, S. Greiner in V. Žumer. History Mechanism Supported Differential Evolution for Chess Evaluation Function Tuning. *Soft Computing – A Fusion of Foundations, Methodologies and Applications*, 15(4):667–682, 2011.

[18] B. Bošković, S. Greiner, J. Brest in V. Žumer. A Differential Evolution for the Tuning of a Chess Evaluation Function. V *The 2006 IEEE Congress on Evolutionary Computation CEC 2006*, strani 6742–6747. IEEE Press, 2006.

[19] B. Brenčič. Optimizacija s čebelami. Diplomska naloga 1. bolonjske stopnje, mentor: J. Brest, somentor: A. Zamuda, Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, 2011.

[20] J. Brest. Constrained Real-Parameter Optimization with ϵ-Self-Adaptive Differential Evolution. V *Constraint-Handling in Evolutionary Optimization*, strani 73–93. Springer, 2009.

[21] J. Brest. Differential Evolution with Self-Adaptation. *Encyclopedia of Artificial Intelligence*, strani 488–493, 2009.

[22] J. Brest, B. Bošković, S. Greiner, V. Žumer in M. S. Maučec. Performance comparison of self-adaptive and adaptive differential evolution algorithms. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 11(7):617–629, 2007.

[23] J. Brest, S. Greiner, B. Bošković, M. Mernik in V. Žumer. Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. *IEEE Transactions on Evolutionary Computation*, 10(6):646–657, 2006.

[24] J. Brest, P. Korošec, J. Šilc, A. Zamuda, B. Bošković in M. S. Maučec. Differential evolution and differential ant-stigmergy on dynamic optimisation problems. *International Journal of Systems Science*, 2012. DOI: 10.1080/00207721.2011.617899.

[25] J. Brest in M. S. Maučec. Population Size Reduction for the Differential Evolution Algorithm. *Applied Intelligence*, 29(3):228–247, 2008.

[26] J. Brest, V. Žumer in M. S. Maučec. Self-adaptive Differential Evolution Algorithm in Constrained Real-Parameter Optimization. V *The 2006 IEEE Congress on Evolutionary Computation CEC 2006*, strani 919–926. IEEE Press, 2006.

[27] J. Brest, V. Žumer in M. S. Maučec. Population size in differential evolution algorithm. *Elektrotehniški vestnik*, 74(1-2):55–60, 2007.

[28] J. Brest, A. Zamuda, B. Bošković, M. S. Maučec in V. Žumer. High-dimensional Real-parameter Optimization Using Self-adaptive Differential Evolution Algorithm with Population Size Reduction. V *2008 IEEE World Congress on Computational Intelligence*, strani 2032–2039. IEEE Press, 2008.

[29] J. Brest, A. Zamuda, B. Bošković, M. S. Maučec in V. Žumer. Dynamic Optimization using Self-Adaptive Differential Evolution. V *IEEE Congress on Evolutionary Computation 2009*, strani 415–422. IEEE Press, 2009.

[30] J. Brest, A. Zamuda, B. Bošković in V. Žumer. Večkriterijska optimizacija: primerjava algoritmov *MOjDE* in *DEMO*. V *Zbornik šestnajste mednarodne Elektrotehniške in računalniške konference ERK 2007, vol. B*, strani 85–88, 2007.

[31] J. Brest, A. Zamuda, B. Bošković in V. Žumer. An Analysis of the Control Parameters' Adaptation in DE. V *Advances in Differential Evolution, Studies in Computational Intelligence*, urednik: U. K. Chakraborty, številka 143, strani 89–110. Springer, 2008.

[32] J. Brest, A. Zamuda, I. Fister, B. Bošković in M. S. Maučec. Constrained real-parameter optimization using a differential evolution algorithm. V *IEEE SSCI2011 symposium series on computational intelligence: proceedings. Piscataway: IEEE*, strani 9–16, 2011.

[33] J. Brest, A. Zamuda, I. Fister in M. S. Maučec. Large Scale Global Optimization using Self-adaptive Differential Evolution Algorithm. V *IEEE World Congress on Computational Intelligence 2010, July 18 - 23, Barcelona, Spain*, strani 3718–3725, 2008.

[34] D. Brown. *Da Vincijeva šifra*. Mladinska knjiga, 2005.

[35] D. Büche, P. Stoll, R. Dornberger in P. Kourmoursakos. Multiobjective Evolutionary Algorithm for the Optimization of Noisy Combustion Processes. *IEEE Transactions on Systems, Man, and Cybernetics Part C—Applications and Reviews*, 32(4):460–473, 2002.

[36] E. A. W. Budge. *The Papyrus of Ani in the British Museum; the Egyptian Text with Interlinear Transliteration and Translation, a Running Translation, Introduction, etc.* British Museum (reprint Dover Publications, 1967), 1895.

[37] A. Caponio, F. Neri in V. Tirronen. Super-fit control adaptation in memetic differential evolution frameworks. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 13(8):811–831, 2009.

[38] C. Chalopin, G. Finet in I. Magnin. Modeling the 3D coronary tree for labeling purposes. *Medical image analysis*, 5(4):301–315, 2001.

[39] B. Chamberlain. *The Kojiki: Records of ancient matters*. Tuttle Publishing, 2005.

[40] X. Chen, B. Neubert, Y.-Q. Xu, O. Deussen in S. B. Kang. Sketch-based tree modeling using markov random field. V *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers*, strani 1–9, New York, NY, USA, 2008. ACM.

[41] N. Chiba, K. Muraoka, A. Doi in J. Hosokawa. Rendering of forest scenery using 3D textures. *Journal of Visualization and Computer Animation*, 8(4):191–199, 1997.

[42] S. Y. Chong, P. Tino in X. Yao. Measuring generalization performance in co-evolutionary learning. *IEEE Transactions on Evolutionary Computation*, 12(4):479–505, 2008.

[43] C. A. Coello Coello. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191(11-12):1245–1287, 2002.

[44] C. A. Coello Coello. 20 Years of Evolutionary Multi-Objective Optimization: What Has Been Done and What Remains to be Done. V *Computational Intelligence: Principles and Practice*, urednika: G. Y. Yen in D. B. Fogel, poglavje 4, strani 73–88. IEEE Computational Intelligence Society, Vancouver, Canada, 2006.

[45] C. A. Coello Coello. Evolutionary multi-objective optimization: a historical view of the field. *Computational Intelligence Magazine, IEEE*, 1(1):28–36, Feb. 2006.

[46] R. L. Cook. Stochastic sampling in computer graphics. *ACM Transactions on Graphics*, 5(1):51–72, 1986.

[47] M. R. T. Dale. *Spatial Pattern Analysis in Plant Ecology*. Cambridge Studies in Ecology. Cambridge University Press, Cambridge, UK, 1999.

[48] C. Darwin. *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. John Murray, 1859.

[49] C. Darwin in A. R. Wallace. On the Tendency of Species to form Varieties; and on the Perpetuation of Varieties and Species by Natural Means of Selection. *Journal of the Proceedings of the Linnean Society of London*, Zoology 3:46—50, 1858.

[50] E. Darwin. *Zoonomia; or, the laws of organic life*, številka 2. Byrne, 1800.

[51] S. Das, A. Abraham, U. Chakraborty in A. Konar. Differential Evolution Using a Neighborhood-based Mutation Operator. *IEEE Transactions on Evolutionary Computation*, 13(3):526–553, 2009.

[52] S. Das in P. N. Suganthan. Differential Evolution: A Survey of the State-of-the-art. *IEEE Transactions on Evolutionary Computation*, 15(1):4–31, 2011.

[53] S. Das in P. N. Suganthan. Problem Definitions and Evaluation Criteria for CEC 2011 Competition on Real World Optimization Problems. Technical report, Dept. of Electronics and Telecommunication Engg., Jadavpur University, India and School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, 2011.

[54] R. Dawkins. *The Selfish Gene – with a new Introduction by the Author*. Oxford University Press, USA, 2006.

[55] J. de la Cruz, B. de Andres-Toro, A. Herrán, E. B. Porta in P. F. Blanco. Multiobjective Optimization of the Transport in Oil Pipelines. V *Proceedings of the 9th IEEE International Conference on Emerging Technologies and Factory Automation*, številka 1, strani 566–573, Lisbon, Portugal, 2003.

[56] K. Deb in H. Gupta. Introducing robustness in multi-objective optimization. *Evolutionary Computation*, 14(4):463–494, 2006.

[57] K. Deb, M. Mohan in S. Mishra. Evaluating the ϵ-Domination Based Multi-Objective Evolutionary Algorithm for a Quick Computation of Pareto-Optimal Solutions. *Evolutionary Computation*, 13(4):501–525, 2005.

[58] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chichester, UK, 2001. ISBN 0-471-87339-X.

[59] K. Deb, S. Agrawal, A. Pratab in T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. V *Proceedings of the Parallel Problem Solving from Nature VI Conference*, uredniki: M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo in H.-P. Schwefel, strani 849–858, Paris, France, 2000. Springer. Lecture Notes in Computer Science No. 1917.

[60] K. Deb, S. Agrawal, A. Pratab in T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. KanGAL report 200001, Indian Institute of Technology, Kanpur, India, 2000.

[61] O. Deussen, C. Colditz, M. Stamminger in G. Drettakis. Interactive visualization of complex plant ecosystems. V *Proceedings of the conference on Visualization 2002*, strani 219–226, 2002.

[62] O. Deussen, P. Hanrahan, B. Lintermann, R. Mech, M. Pharr in P. Prusinkiewicz. Realistic modeling and rendering of plant ecosystems. V *Proceedings of SIGGRAPH '98*, strani 275–286, 1998.

[63] F. di Pierro, S.-T. Khu in D. A. Savić. An Investigation on Preference Order-Ranking Scheme for Multiobjective Evolutionary Optimization. *IEEE Transactions on Evolutionary Computation*, 11(1):17–45, 2007.

[64] DigiBen. GameTutorials OpenGL Tutorials, 2005. URL `http://www.gametutorials.com/gtstore/c-1-test-cat.aspx`; dostop 26. may 2010.

[65] J. Dobry. OpenGL - Getting High Performance, 2000. URL `http://jdobry.webpark.cz/opengl/opengl_maximum_performance.html`; dostop 4. maj 2006.

[66] M. Dorigo, V. Maniezzo in A. Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 26(1):29–41, 1996.

[67] W. Duch in J. Mandziuk. Quo Vadis Computational Intelligence? V *Machine Intelligence: Quo Vadis?*, urednik: K. H. P. Sincak, J. Vascak, številka 21 iz *Advances in Fuzzy Systems – Applications and Theory*, strani 3–28. World Scientific, 2004.

[68] P. Ehrenfreund, W. Irvine, L. Becker, J. Blank, J. R. Brucato, L. Colangeli, S. Derenne, D. Despois, A. Dutrey, H. Fraaije, A. Lazcano, T. Owen, F. Robert in I. S. S. I. ISSI-Team. Astrophysical and astrochemical insights into the origin of life. *Reports on Progress in Physics*, 65(10):1427, 2002.

[69] A. E. Eiben in J. E. Smith. *Introduction to Evolutionary Computing (Natural Computing Series)*. Springer, 2003.

[70] A. E. Eiben, R. Hinterding in Z. Michalewicz. Parameter Control in Evolutionary Algorithms. *IEEE Trans. on Evolutionary Computation*, 3(2):124–141, 1999.

[71] A. Einstein. Die grundlage der allgemeinen relativitätstheorie. *Annalen der Physik*, 354(7):769–822, 1916.

[72] P. Engrand. A multi-objective optimization approach based on simulated annealing and its application to nuclear fuel management. V *Proceedings of the Fifth International Conference on Nuclear Engineering*, strani 416–423, Nice, France, 1997. American Society of Mechanical Engineering.

[73] H.-Y. Fan in J. Lampinen. A Trigonometric Mutation Operation to Differential Evolution. *Journal of Global Optimization*, 27(1):105–129, 2003.

[74] V. Feoktistov. *Differential Evolution: In Search of Solutions (Springer Optimization and Its Applications)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[75] A. Feragen, F. Lauze, P. Lo, M. de Bruijne in M. Nielsen. Geometries on spaces of treelike shapes. V *Computer Vision – ACCV 2010*, uredniki: R. Kimmel, R. Klette in A. Sugimoto, številka 6493 iz *Lecture Notes in Computer Science*, strani 160–173. Springer Berlin, Heidelberg, 2011.

[76] D. B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, Third Edition (IEEE Press Series on Computational Intelligence)*. Wiley-IEEE Press, Hoboken, New Jersey, 2006.

[77] L. J. Fogel, A. J. Owens in M. J. Walsh. *Artificial Intelligence Through Simulated Evolution*. John Wiley & Sons Inc, 1966.

[78] A. Fraser. Simulation of Genetic Systems by Automatic Digital Computers VI. Epistasis. *Australian Journal of Biological Sciences*, 13(2):150–162, 1957.

[79] T. Freeth, Y. Bitsakis, X. Moussas, J. Seiradakis, A. Tselikas, H. Mangou, M. Zafeiropoulou, R. Hadland, D. Bate, A. Ramsey et al. Decoding the ancient greek astronomical calculator known as the antikythera mechanism. *Nature*, 444(7119):587, 2006.

[80] Z. Geem in J. Choi. Music composition using harmony search algorithm. *Applications of Evolutionary Computing*, strani 593–600, 2007.

[81] S. Ghosh, S. Roy, S. Islam, S. Das in P. Suganthan. A differential covariance matrix adaptation evolutionary algorithm for global optimization. V *Differential Evolution (SDE), 2011 IEEE Symposium on*, strani 1 –8, 2011.

[82] F. Glover. Tabu Search – Part I. *ORSA Journal on computing*, 1(3):190–206, 1989.

[83] C. K. Goh in K. C. Tan. An Investigation on Noisy Environments in Evolutionary Multiobjective Optimization. *IEEE Transactions on Evolutionary Computation*, 11(3):354–381, 2007.

[84] D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys (CSUR)*, 23(1):5–48, 1991.

[85] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.

[86] G. Gribb in K. Hartmann. Fast extraction of viewing frustum planes from the world-view-projection matrix, 2001. URL `http://www2.ravensoft.com/users/ggribb/planeextraction.pdf`; dostop 27. marec 2006.

[87] N. Guid. *Računalniška grafika*. Fakulteta za elektrotehniko, računalništvo in informatiko, Univerza v Mariboru, 2001.

[88] Y. Y. Haimes, L. S. Lasdon in D. A. Wismer. On a bicriterion formulation of the problems of integrated system identification in system optimization. *IEEE Transactions on Systems, Man in Cybernetics*, 1(3):296–297, 1971.

[89] M. P. Hansen in A. Jaszkiewicz. Evaluating the quality of approximations to the non-dominated set. Technical Report IMM-REP-1998-7, Technical University of Denmark, 1998.

[90] N. Hansen, S. Muller in P. Koumoutsakos. Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evolutionary Computation*, 11(1):1–18, 2003.

[91] N. Hansen in A. Ostermeier. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[92] R. P. Hardie in R. K. Gaye. *English translation of Aristotle: Physics, 350 AC.* Oxford: The Clarendon Press, 1930.

[93] S. Hawking, C. Sagan, R. Miller et al. *A brief history of time: From the big bang to black holes.* Bantam Books, 1988.

[94] S. Heath. *The Method of Archimedes, recently discovered by Heiberg: A supplement to the Works of Archimedes.* Cosimo Classics, 2007.

[95] J. Holland. *Adaptation In Natural and Artificial Systems.* The University of Michigan Press, Ann Arbor, 1975.

[96] M. Holton. Strands, gravity, and botanical tree imagery. *Comput. Graph. Forum*, 13(1):57–67, 1994.

[97] B. Hopkins. A new method for determining the type of distribution of plant individuals. *Annals of Botany*, XVIII:213–226, 1954.

[98] V. L. Huang, A. K. Qin, K. Deb, E. Zitzler, P. N. Suganthan, J. J. Liang, M. Preuss in S. Huband. Problem Definitions for Performance Assessment & Competition on Multi-objective Optimization Algorithms. Technical Report TR-07-01, Nanyang Technological University et. al., Singapore, 2007.

[99] V. L. Huang, A. K. Qin in P. N. Suganthan. Self-adaptive Differential Evolution Algorithm for Constrained Real-Parameter Optimization. V *2006 Congress on Evolutionary Computation (CEC 2006)*, strani 17–24. IEEE Service Center, 2006.

[100] V. L. Huang, A. K. Qin, P. N. Suganthan in M. F. Tasgetiren. Multi-objective Optimization based on Self-adaptive Differential Evolution. V *The 2007 IEEE Congress on Evolutionary Computation CEC 2007*, strani 3601–3608. IEEE Press, 2007.

[101] V. L. Huang, P. N. Suganthan, A. K. Qin in S. Baskar. Multiobjective Differential Evolution with External Archive and Harmonic Distance-Based Diversity Measure. Technical Report TR-07-01, Nanyang Technological University, Singapore, 2006.

[102] E. J. Hughes. Swarm Guidance using a Multi-Objective Co-evolutionary On-Line Evolutionary Algorithm. V *2004 Congress on Evolutionary Computation (CEC 2004)*, številka 2, strani 2357–2363, Portland, Oregon, USA, 2004. IEEE Service Center.

[103] J. Huxley. *Evolution: The Modern Synthesis.* Allen & Unwin, London, 1942.

[104] J. Huxley. The emergence of Darwinism. *Perspectives in biology and medicine*, 3:321, 1960.

[105] T. H. Huxley. ART. VIII. - Darwin on the origin of Species. *Westminster Review*, 3:541–570, 1960.

[106] J. Hyyppä, H. Hyyppä, D. Leckie, F. Gougeon, X. Yu in M. Maltamo. Review of methods of small-footprint airborne laser scanning for extracting forest inventory data in boreal forests. *International Journal of Remote Sensing*, 29(5):1339–1366, 2008.

[107] C. Igel, N. Hansen in S. Roth. Covariance Matrix Adaptation for Multi-objective Optimization. *Evolutionary Computation*, 15(1):1–28, 2007.

[108] W. P. in drugi. Wikipedia, the free encyclopedia. V *Wikimedia Foundation*, 2011. URL `http://en.wikipedia.org/`.

[109] A. W. Iorio in X. Li. Incorporating Directional Information within a Differential Evolution Algorithm for Multi-objective Optimization. V *2006 Genetic and Evolutionary Computation Conference (GECCO 2006)*, urednik: M. K. et al., številka 1, strani 691–697, Seattle, Washington, USA, 2006. ACM Press. ISBN 1-59593-186-4.

[110] W. D. Jones. How to build a mile-high skyscraper. *Spectrum, IEEE*, 44(6):52–53, 2007.

[111] R. Joshi in A. Sanderson. Minimal representation multisensor fusion using differential evolution. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 29(1):1083–4427, 1999.

[112] D. Karaboga in B. Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of Global Optimization*, 39(3):459–471, 2007.

[113] J. Kennedy in R. Eberhart. Particle swarm optimization. V *Neural Networks, 1995. Proceedings., IEEE International Conference on*, številka 4, strani 1942–1948. IEEE, 1995.

[114] J. Kephart. A biologically inspired immune system for computers. V *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, strani 130–139, 1994.

[115] S. Kirkpatrick, C. Gelatt in M. Vecchi. Optimization by simulated annealing. *science*, 220(4598):671, 1983.

[116] M. Kitti. History of Optimization. V *Systeemianalyysin laboratorio*, 3. 10. 2011. URL `http://www.mitrikitti.fi/opthist.html`.

[117] J. Klein. Breve: a 3D environment for the simulation of decentralized systems and artificial life. *Artificial life eight*, stran 329, 2003.

[118] J. Knowles, L. Thiele in E. Zitzler. A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers. 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland, 2006. revised version.

[119] D. Knuth. Computer programming as an art. *Communications of the ACM*, 17(12):667–673, 1974.

[120] P. Korošec, J. Šilc in B. Filipič. The differential ant-stigmergy algorithm. *Information Sciences*, 2012. DOI: 10.1016/j.ins.2010.05.002.

[121] P. Korošec in J. Šilc. The continuous differential ant-stigmergy algorithm applied to bound constrained real-world optimization problem. V *The 2011 IEEE Congress on Evolutionary Computation (CEC 2011)*, strani 1327–1334, New Orelans, USA, June 5-8 2011.

[122] P. Korošec, J. Šilc, K. Oblak in F. Kosel. The Differential Ant-Stigmergy Algorithm: An Experimental Evaluation and a Real-World Application. V *The 2007 IEEE Congress on Evolutionary Computation CEC 2007*, strani 157–164. IEEE Press, 2007.

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

[124] S. Koziel in Z. Michalewicz. Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization. *Evolutionary Computation*, 7(1):19–44, 1999.

[125] S. Kukkonen in J. Lampinen. GDE3: The third Evolution Step of Generalized Differential Evolution. V *2005 IEEE Congress on Evolutionary Computation (CEC 2005)*, številka 1, strani 443–450, Edinburgh, Scotland, 2005. IEEE Service Center.

[126] S. Kukkonen in J. Lampinen. Performance Assessment of Generalized Differential Evolution 3 (GDE3) with a Given Set of Problems. V *The 2007 IEEE Congress on Evolutionary Computation CEC 2007*, strani 3593–3600. IEEE Press, 2007.

[127] A. Kumar, D. Sharma in K. Deb. A Hybrid Multi-Objective Optimization Procedure Using PCX Based NSGA-II and Sequential Quadratic Programming. V *The 2007 IEEE Congress on Evolutionary Computation CEC 2007*, strani 3011–3018. IEEE Press, 2007.

[128] D. Kundu, K. Suresh, S. Ghosh, S. Das, B. K. Panigrahi in S. Das. Multi-objective optimization with artificial weed colonies. *Information Sciences*, 181(12):2441–2454, 2011.

[129] J. Lampinen. A Bibliography of Differential Evolution Algorithm. Technical report, Lappeenranta University of Technology, Department of Information Technology, Laboratory of Information Processing, 2001. URL `http://www2.lut.fi/~jlampine/debiblio.htm`.

[130] L. Lamport. *LATEX: A Document Preparation System*. Addison-Wesley, Reading, Massachusetts, second edition, 1994.

[131] B. Lane in P. Prusinkiewicz. Generating spatial distributions for multilevel models of plant communities. V *Proceedings of the Graphics Interface 2002 (GI-02)*, strani 69–80, Mississauga, Ontario, Canada, may 27–29 2002. Canadian Information Processing Society.

[132] W. Langdon, S. Gustafson in J. Koza. The genetic programming bibliography. V *Collection of Computer Science Bibliographies*, 2011. URL `http://www.cs.bham.ac.uk/~wbl/biblio/`.

[133] M. Laumanns, L. Thiele in E. Zitzler. An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. *European Journal of Operational Research*, 169(3):932–942, 2006.

[134] C. T. Lawrence, J. L. Zhou in A. L. Tits. User's Guide for CFSQP Version 2.5: A C Code for Solving (Large Scale) Constrained Nonlinear (Minimax) Optimization Problems, Generating Iterates Satisfying All Inequality Constraints. Technical report, Institute for Systems Research, University of Maryland, College Park, MD 20742, 1997. TR-94-16r1.

[135] F. Lewis. *Applied optimal control and estimation*. Prentice Hall PTR, 1992.

[136] Z. Li, Q. Zhu in C. Gold. *Digital Terrain Modeling*. CRC Press, San Francisco, 2005.

[137] J. J. Liang, T. P. Runarsson, E. Mezura-Montes, M. Clerc, P. N. Suganthan, C. A. C. Coello in K. Deb. Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization. Technical Report Technical Report Report 2006005, Nanyang Technological University, Singapore and et al., 2005.

[138] K. Liang, X. Yao in C. Newton. Adapting Self-adaptive Parameters in Evolutionary Algorithms, 2001.

[139] E. Limpert, W. Stahel in M. Abbt. Log-normal Distributions across the Sciences: Keys and Clues. *BioScience*, 51(5):341–352, 2001.

[140] A. Lindenmayer. Mathematical models for cellular interactions in development, I & II. *J. Theor. Biol.*, 18:280–315, 1968.

[141] C. Linnæ us. *Systema naturœper regna tria naturæ, secundum classes, ordines, genera, species, cum characteribus, differentiis, synonymis, locis*. Laurentii Salvii, Stockholm, 1758.

[142] B. Lintermann in O. Deussen. Interactive modeling of plants. *IEEE Computer Graphics and Applications*, 19(1):56–65, 1999.

[143] J. Liu in J. Lampinen. A Fuzzy Adaptive Differential Evolution Algorithm. *Soft Comput.*, 9(6):448–462, 2005.

[144] M. Livio. *The golden ratio: The story of phi, the world's most astonishing number*. Broadway, 2003.

[145] N. K. Madavan. Multiobjective Optimization Using a Pareto Differential Evolution Approach. V *Congress on Evolutionary Computation*, številka 2, strani 1145–1150, Piscataway, New Jersey, 2002. IEEE Service Center.

[146] R. Mallipeddi in P. N. Suganthan. Ensemble Differential Evolution Algorithm for CEC2011 Problems. V *The 2011 IEEE Congress on Evolutionary Computation (CEC 2011)*, stran 68. IEEE Press, 2011.

[147] R. Mallipeddi, P. N. Suganthan, Q. K. Pan in M. F. Tasgetiren. Differential evolution algorithm with ensemble of parameters and mutation strategies. *Applied Soft Computing*, 11(2):1679–1696, 2011.

[148] B. Mandelbrot. *The fractal geometry of nature*. Freeman, San Francisco, 1982.

[149] G. Marsaglia. The structure of linear congruential sequences. V *Appl. Number Theory numer. Analysis, Proc. Sympos. Univ. Montreal 1971, 249–285*, urednik: S. K. Zaremba, strani 249–285. Academic Press, New York, NY, USA, 1972.

[150] A. Martinčič, D. Vrhovšek in F. Batič. *Ekologija rastlin*. VTO za biologijo, Pleško, Ljubljana, Rožna dolina c. IV/36, 1981.

[151] U. Maulik in I. Saha. Modified differential evolution based fuzzy clustering for pixel classification in remote sensing imagery. *Pattern Recognition*, 42(9):2135–2149, 2009.

[152] J. McCarthy. Recursive functions of symbolic expressions and their computation by machine, Part I. *Communications of the ACM*, 3(4):184–195, 1960.

[153] J. McCarthy, M. Minsky, N. Rochester in C. Shannon. A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, August 31, 1955. *AI Magazine*, 27(4):12, 2006.

[154] R. Mech in P. Prusinkiewicz. Visual models of plants interacting with their environment. V *Proceedings of SIGGRAPH '96*, strani 397–410, 1996.

[155] G. Mendel. *Experiments in Plant Hybridization*. Cosimo, Inc., 1865.

[156] E. Mezura-Montes. *Alternative Techniques to Handle Constraints in Evolutionary Optimization*. Doktorska disertacija, Computer Science Section, Electrical Eng. Department, 2004.

[157] E. Mezura-Montes in B. C. Lopez-Ramirez. Comparing bio-inspired algorithms in constrained optimization problems. *The 2007 IEEE Congress on Evolutionary Computation*, strani 662–669, 25-28 Sept. 2007.

[158] E. Mezura-Montes, J. Velázquez-Reyes in C. A. C. Coello. A comparative study of differential evolution variants for global optimization. V *GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, strani 485–492, New York, NY, USA, 2006. ACM Press.

[159] Z. Michalewicz in D. B. Fogel. *How to Solve It: Modern Heuristics*. Springer, Berlin, 2000.

[160] Z. Michalewicz in M. Schoenauer. Evolutionary Algorithms for Constrained Parameter Optimization Problems. *Evolutionary Computation*, 4(1):1–32, 1996.

[161] E. Mininno, F. Neri, F. Cupertino in D. Naso. Compact Differential Evolution. *IEEE Transactions on Evolutionary Computation*, 15(1):32–54, 2011.

[162] T. I. Museum. Kumranski rokopisi. V *The Digital Dead Sea Scrolls*, Digitalizirano (Google), 2011. URL `http://dss.collections.imj.org.il/`.

[163] J. Nelder in R. Mead. A simplex method for function minimization. *The computer journal*, 7(4):308, 1965.

[164] F. Neri in C. Cotta. Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation*, Objavljeno online 30. november 2011.

[165] F. Neri in E. Mininno. Memetic compact differential evolution for cartesian robot control. *IEEE Computational Intelligence Magazine*, 5(2):54–65, 2010.

[166] F. Neri in V. Tirronen. Recent Advances in Differential Evolution: A Survey and Experimental Analysis. *Artificial Intelligence Review*, 33(1–2):61–106, 2010.

[167] B. Neubert, T. Franken in O. Deussen. Approximate image-based tree-modeling using particle flows. *ACM Trans. Graph.*, 26(3):88, 2007.

[168] I. Newton. Philosophiæ naturalis principia mathematica. *Royal Society, London*, 1687.

[169] E. Nisbet in N. Sleep. The habitat and nature of early life. *Nature*, 409(6823):1083–1091, 2001.

[170] J. J. O'Connor in E. F. Robertson. A chronology of pi. Technical report, The MacTutor History of Mathematics archive, 2000. URL `http://www-history.mcs.st-andrews.ac.uk/HistTopics/Pi_chronology.html`.

[171] J. J. O'Connor in E. F. Robertson. George dantzig. Technical report, The MacTutor History of Mathematics archive, 2003. URL `http://www-history.mcs.st-andrews.ac.uk/Biographies/Dantzig_George.html`.

[172] M. G. Omran, A. Salman in A. P. Engelbrecht. Self-adaptive Differential Evolution. V *Computational intelligence and security*, strani 192–199, 2005.

[173] P. E. Oppenheimer. Real time design and animation of fractal plants and trees. *Computer Graphics*, 20(4):55–64, 1986.

[174] S. Papert, D. Watt, A. diSessa in S. Weir. Final report of the brookline LOGO project. Technical Report 545, MIT AI Lab, 1979.

[175] L. Pareto. *Zgodovina človeštva - Stari svet, II/I.* DZS, Ljubljana, str. 251, 1970.

[176] K. Passino. Bacterial foraging optimization. *International Journal of Swarm Intelligence Research*, 1(1):1–16, 2010.

[177] D. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim in M. Zaidi. The Bees Algorithm – A Novel Tool for Complex Optimisation Problems. V *Proceedings of IPROMS 2006 conference*, strani 454–461, 2006.

[178] J. Phattaralerphong in H. Sinoquet. A method for 3d reconstruction of tree crown volume from photographs: assessment with 3d-digitized plants. *Tree Physiology*, 25(10):1229–1242, 2005.

[179] D. Picco. Frustum culling, 2003. URL `http://www.flipcode.com/articles/article_frustumculling-pf.shtml`; dostop 27. marec 2006.

[180] A. Posamentier in N. Gordan. An astounding revelation on the history of pi. *Mathematics Teacher*, 1984.

[181] O. E. Pressman. Terrain engine, 2001. URL `http://ohad.visual-i.com/exper/exper.htm`.

[182] K. Price in R. Storn. Differential Evolution: A Simple Evolution Strategy for Fast Optimization. *Dr. Dobb's Journal of Software Tools*, 22(4):18–24, 1997.

[183] K. V. Price, R. M. Storn in J. A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Natural Computing Series. Springer-Verlag, Berlin, Germany, 2005.

[184] W. Price. A controlled random search procedure for global optimisation. *The Computer Journal*, 20(4):367–370, 1977.

[185] P. Prusinkiewicz, M. Hammel, R. M. ech in J. Hanan. The artificial life of plants. V *Artificial Life for Graphics, Animation, and Virtual Reality*, številka 7 iz *SIGGRAPH'95 Course Notes*, strani 1–38. ACM, ACM Press, 1995.

[186] P. Prusinkiewicz, M. Hammel, J. Hanan in R. Měch. L-systems: From the theory to visual models of plants. V *Plants to Ecosystems*, urednik: M. T. Michalewicz, številka 1 iz *Advances in Computational Life Sciences*, poglavje 1, strani 1–27. CSIRO Publishing, P.O. Box 1139, Collingwood 3066, Australia, 1997.

[187] P. Prusinkiewicz in A. Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag, 1990.

[188] A. K. Qin, V. L. Huang in P. N. Suganthan. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, 13(2):398–417, 2009.

[189] A. K. Qin in P. N. Suganthan. Self-adaptive Differential Evolution Algorithm for Numerical Optimization. V *Proceedings of the 2005 Congress on Evolutionary Computation*, številka 2, strani 1785–1791. IEEE Press, 2005.

[190] L. Quan. *Image-Based Modeling*. Springer Publishing Company, Incorporated, 1st edition, 2010.

[191] L. Quintana-Murci, O. Semino, H. Bandelt, G. Passarino, K. McElreavey in A. Santachiara-Benerecetti. Genetic evidence of an early exit of Homo sapiens sapiens from Africa through eastern Africa. *Nat Genet*, 23(4):437–441, 1999.

[192] T. S. Ray. Evolution and optimization of digital organisms. *Scientific excellence in supercomputing: The IBM*, strani 489–531, 1990.

[193] A. Reche, I. Martin in G. Drettakis. Volumetric reconstruction and interactive rendering of trees from photographs. V *ACM Transactions on Graphics (SIGGRAPH) Conference Proceedings*, strani 720–727, 2004.

[194] I. Rechenberg. *Evolutionsstrategie–Optimierung technisher Systeme nach Prinzipien der biologischen Evolution*. Doktorska disertacija, Frommann-Holzboog, Stuttgart, 1973.

[195] W. Reeves. Approximate and probabilistic algorithms for shading and rendering structured particle systems. *Proceedings of SIGGRAPH'85*, strani 313–322, 1985.

[196] D. Reinhardt, E.-R. Pesce, P. Stieger, T. Mandel, K. Baltensperger, M. Bennett, J. Traas, J. Friml in C. Kuhlemeier. Regulation of phyllotaxis by polar auxin transport. *Nature*, 426(6964):255–260, 2003.

[197] G. V. Reklaitis, A. Ravindran in K. M. Ragsdell. *Engineering Optimization: Methods and Applications*. Wiley-Interscience, 1983.

[198] R. E. Ricklefs. *Ecology*. W. H. Freeman, New York, third edition, 1990.

[199] Rigveda. *Vede*. Indija, 2000 AC.

[200] T. Robič. Performance of DEMO on New Test Problems: A Comparison Study. V *Proceedings of the Fourteenth International Electrotechnical and Computer Science Conference – ERK 2005*, številka B, strani 121–124, 2005.

[201] T. Robič in B. Filipič. DEMO: Differential Evolution for Multiobjective Optimization. V *Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization – EMO 2005*, številka 3410 iz *Lecture Notes in Computer Science*, strani 520–533. Springer, 2005.

[202] G. Romanes. Post-darwinian questions: heredity and utility. V *Darwin, and after Darwin*. Longmans, Green: London, 1895.

[203] A. Runions, B. Lane in P. Prusinkiewicz. Modeling trees with a space colonization algorithm. *Prague, Czech Republic*, strani 63–70, 2007.

[204] A. Samal, J. Brandle in D. Zhang. Texture as the basis for individual tree identification. *Information Sciences*, 176(5):565–576, 2006.

[205] L. V. Santana-Quintero in C. A. Coello Coello. An Algorithm Based on Differential Evolution for Multi-Objective Problems. *International Journal of Computational Intelligence Research*, 1(2):151–169, 2005.

[206] J. D. Schaffer. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. V *Proceedings of the 1st International Conference on Genetic Algorithms*, strani 93–100, Mahwah, NJ, USA, 1985. Lawrence Erlbaum Associates, Inc.

[207] H.-P. Schwefel. *Evolution and Optimum Seeking*. Sixth-Generation Computer Technology. Wiley Interscience, New York, 1995.

[208] D. Sharma, A. Kumar, K. Deb in K. Sindhya. Hybridization of SBX Based NSGA-II and Sequential Quadratic Programming for Solving Multi-objective Optimization Problems. V *The 2007 IEEE Congress on Evolutionary Computation CEC 2007*, strani 3003–3010. IEEE Press, 2007.

[209] K. J. Shaw, A. L. Nortcliffe, M. Thompson, J. Love, C. M. Fonseca in P. J. Fleming. Assessing the Performance of Multiobjective Genetic Algorithms for Optimization of a Batch Process Scheduling Problem. V *1999 Congress on Evolutionary Computation*, strani 37–45, Washington, D.C., July 1999. IEEE Service Center.

[210] M. Smid. Closest-Point Problems in Computational Geometry. *Handbook on Computational Geometry, edited by J.-R. Sack, North Holand, Amsterdam*, strani 877–935, 2000.

[211] K. Sone, A. Suzuki, S. Miyazawa, K. Noguchi in I. Terashima. Maintenance mechanisms of the pipe model relationship and Leonardo da Vinci's rule in the branching architecture of Acer rufinerve trees. *Journal of plant research*, 122(1):41–52, 2009.

[212] N. Srinivas in K. Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

[213] A. P. Stakhov. The golden section in the measurement theory. *Computers & Mathematics with Applications*, 17(4-6):613–638, 1989.

[214] R. Storn in K. Price. Differential Evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, Berkeley, CA, 1995.

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

[216] D. Strnad in N. Guid. Modeling trees with hypertextures. *Comput. Graph. Forum*, 23(2):173–188, 2004.

[217] T. Takahama, S. Sakai in N. Iwane. Solving Nonlinear Constrained Optimization Problems by the ϵ Constrained Differential Evolution. *IEEE International Conference on Systems, Man and Cybernetics 2006 (SMC 2006)*, 3:2322–2327, 2006.

[218] K. Tan, Y. Yang in C. Goh. A distributed Cooperative coevolutionary algorithm for multiobjective optimization. *Evolutionary Computation, IEEE Transactions on*, 10(5):527–549, 2006.

[219] K. Tang. Summary of Results on CEC'08 Competition on Large Scale Global Optimization. Technical report, Nature Inspired Computation and Applications Laboratory, USTC, China, URL `http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC-08/CEC2008_SUMMARY.pdf`, 2008.

[220] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y. P. Chen, C. M. Chen in Z. Yang. Benchmark Functions for the CEC 2008 Special Session and Competition on Large Scale Global Optimization. Technical report, Nature Inspired Computation and Applications Laboratory et. al., USTC, China, URL `http://nical.ustc.edu.cn/cec08ss.php`, 2007.

[221] J. Teo. Exploring dynamic self-adaptive populations in differential evolution. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 10(8):673–686, 2006.

[222] J. H. Tigay. *The evolution of the Gilgamesh epic*. University of Pennsylvania Press, 1982.

[223] V. Tirronen, F. Neri, T. Kärkkäinen, K. Majava in T. Rossi. An enhanced memetic differential evolution in filter design for defect detection in paper production. *Evolutionary Computation*, 16(4):529–555, 2008.

[224] L.-Y. Tseng in C. Chen. Multiple Trajectory Search for Multiobjective Optimization. V *The 2007 IEEE Congress on Evolutionary Computation CEC 2007*, strani 3609–3616. IEEE Press, 2007.

[225] A. Turing. Intelligent machinery (1948). *The essential Turing: seminal writings in computing, logic, philosophy, artificial intelligence, and artificial life, plus the secrets of Enigma*, stran 395, 2004.

[226] A. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

[227] T. Tušar in B. Filipič. Differential Evolution versus Genetic Algorithms in Multiobjective Optimization. V *Proceedings of the Fourth International Conference on Evolutionary Multi-Criterion Optimization – EMO 2007*, številka 4403 iz *Lecture Notes in Computer Science*, strani 257–271. Springer, 2007.

[228] T. Tušar, P. Korošec, G. Papa, B. Filipič in J. Šilc. A comparative study of stochastic optimization methods in electric motor design. *Applied Intelligence*, 2(27):101–111, 2007.

[229] T. Tušar. Design of an Algorithm for Multiobjective Optimization with Differential Evolution. M.Sc. Thesis, Faculty of Computer and Information Science, University of Ljubljana, 2007.

[230] J. Tvrdík. Adaptation in differential evolution: A numerical comparison. *Applied Soft Computing*, 9(3):1149–1155, 2009.

[231] M. P. van Lieshout. Time dependent energy calculations. V *Wind flow and Trees*. British Wind Energy Association, 2004.

[232] E. Vera. *Bryce 5 User Guide*. MetaCreations Corporation, 2001.

[233] S. von Mammen in C. Jacob. Genetic Swarm Grammar Programming: Ecological Breeding Like a Gardener. V *2007 IEEE Congress on Evolutionary Computation*, urednika: D. Srinivasan in L. Wang, strani 851–858, Singapore, 25-28 September 2007. IEEE Computational Intelligence Society, IEEE Press.

[234] S. von Mammen in C. Jacob. The Evolution of Swarm Grammars: Growing Trees, Crafting Art and Bottom-Up Design. *IEEE Computational Intelligence Magazine*, 4(3):10–19, 2009.

[235] A. R. Wallace. On the Tendency of Species to form Varieties. *Journal of the Proceedings of the Linnean Society of London*, Zoology 3:53—62, 1858.

[236] S. Watanabe, T. Hiroyasu in M. Miki. Parallel Evolutionary Multi-Criterion Optimization for Mobile Telecommunication Networks Optimization. V *Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems. Proceedings of the EUROGEN 2001. Athens. Greece, September 19-21*, uredniki: K. Giannakoglou, D. Tsahalis, J. Periaux, K. Papailiou in T. Fogarty, strani 167–172, Barcelona, Spain, 2001. International Center for Numerical Methods in Engineering (CIMNE).

[237] J. Weber in J. Penn. Creation and rendering of realistic trees. *Proceedings of SIGGRAPH '95*, strani 119–128, 1995.

[238] M. Weber, F. Neri in V. Tirronen. A Study on Scale Factor in Distributed Differential Evolution. *Information Sciences*, 181(12), 2011.

[239] A. Weismann. *The Germ-Plasm: A Theory of Heredity*. Charles Scribner's Sons, 1893.

[240] Wikipedia. OpenGL — Wikipedia, The Free Encyclopedia, 2006. URL `http://en.wikipedia.org/w/index.php?title=OpenGL&oldid=92006684`.

[241] J. Wojtusiak in R. Michalski. The lem3 implementation of learnable evolution model and its testing on complex function optimization problems. V *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, strani 1281–1288. ACM, 2006.

[242] M. Woo et al. *OpenGL programming guide: the official guide to learning OpenGL, version 1.2*. Addison-Wesley, third edition, 1999.

[243] F. Xue, A. C. Sanderson in R. J. Graves. Pareto-based Multi-Objective Differential Evolution. V *Proceedings of the 2003 Congress on Evolutionary Computation*, številka 2, strani 862–869, Canberra, Australia, 2003. IEEE Press.

[244] X. Yang. Firefly algorithm, stochastic test functions and design optimisation. *International Journal of Bio-Inspired Computation*, 2(2):78–84, 2010.

[245] X. Yang in S. Deb. Cuckoo search via Lévy flights. V *NaBIC 2009 World Congress on Nature & Biologically Inspired Computing*, strani 210–214. IEEE, 2009.

[246] Z. Yang, J. He in X. Yao. Making a Difference to Differential Evolution. V *Advances in Metaheuristics for Hard Optimization*, urednika: Z. Michalewicz in P. Siarry, Lecture Notes in Computer Science, strani 397–414, Berlin, 2008. Springer.

[247] Z. Yang, K. Tang in X. Yao. Differential Evolution for High-Dimensional Function Optimization. V *Proceedings of the 2007 IEEE Congress on Evolutionary Computation CEC 2007*, strani 3523–3530, Singapore, 25-28 September 2007.

[248] X. Yao, Y. Liu in G. Lin. Evolutionary Programming Made Faster. *IEEE Transactions on Evolutionary Computation*, 3(2):82–102, 1999.

[249] O. Yogev, A. A. Shapiro in E. K. Antonsson. Computational Evolutionary Embryogeny. *IEEE Transactions on Evolutionary Computation*, 14(2):301–325, 2010.

[250] E. L. Yu in P. N. Suganthan. Ensemble of niching algorithms. *Information Sciences*, 180(15), 2010.

[251] D. Zaharie. Influence of crossover on the behavior of Differential Evolution Algorithms. *Applied Soft Computing*, 9(3):1126–1138, 2009.

[252] K. Zakšek, K. Oštir in T. Podobnikar. Osončenost površja slovenije. *Geografski vestnik*, 76(1):79–90, 2004.

[253] E. N. Zalta (glavni urednik). The Stanford Encyclopedia of Philosophy. V *The Metaphysics Research Lab, Center for the Study of Language and Information, Stanford University, Stanford, CA 94305-4115*, 2011. URL `http://plato.stanford.edu/`.

[254] A. Zamuda. Interaktivno gojenje ekosistemov z operatorji evolucije. V *FILIPIČ, Bogdan (ur.), MERNIK, Marjan (ur.), PAPA, Gregor (ur.). Zbornik povzetkov delavnic Algoritmi po vzorih iz narave v študijskem letu 2007/2008: Ljubljana, september 2008. Ljubljana: Institut Jožef Stefan*, stran 11, 2007.

[255] A. Zamuda. Samoprilagajanje krmilnih parametrov pri algoritmu diferencialne evolucije za večkriterijsko optimizacijo. Mag. naloga, Fakulteta za elektrotehniko, računalništvo in informatiko, Univerza v Mariboru, 2008.

[256] A. Zamuda. Diferencialna evolucija realnih industrijskih izzivov CEC 2011. V *Zbornik dvajsete mednarodne Elektrotehniške in računalniške konference ERK 2011, vol. B*, strani 185–188, 2011.

[257] A. Zamuda in J. Brest. Večkriterijska rekonstrukcija numerično kodiranih proceduralnih modelov dreves z diferencialno evolucijo. V *Zbornik devetnajste mednarodne Elektrotehniške in računalniške konference ERK 2010, vol. B*, strani 169–172, 2010.

[258] A. Zamuda in J. Brest. Population Reduction Differential Evolution with Multiple Mutation Strategies in Real World Industry Challenges. V *Swarm and Evolutionary Computation*, uredniki: L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. Zadeh in J. Zurada, Lecture Notes in Computer Science, strani 154–161. Springer Berlin / Heidelberg, 2012.

[259] A. Zamuda in J. Brest. Tree Model Reconstruction Innovization Using Multi-objective Differential Evolution. V *2012 IEEE World Congress on Computational Intelligence (IEEE WCCI 2012)*, Brisbane, Australia, 2012. IEEE Press.

[260] A. Zamuda, J. Brest, B. Bošković in V. Žumer. Differential Evolution for Multiobjective Optimization with Self Adaptation. V *The 2007 IEEE Congress on Evolutionary Computation CEC 2007*, strani 3617–3624. IEEE Press, 2007.

[261] A. Zamuda, J. Brest, B. Bošković in V. Žumer. Večkriterijska optimizacija: eksperimentalni rezultati algoritmov *MOjDE* in *DEMO*. V *Zbornik šestnajste mednarodne Elektrotehniške in računalniške konference ERK 2007, vol. B*, strani 89–92, 2007.

[262] A. Zamuda, J. Brest, B. Bošković in V. Žumer. Diferencialna evolucija za večkriterijsko optimizacijo s samoprilagajanjem in z lokalnim preiskovanjem SQP. V *Zbornik sedemnajste mednarodne Elektrotehniške in računalniške konference ERK 2008, vol. B*, strani 103–106, 2008.

[263] A. Zamuda, J. Brest, B. Bošković in V. Žumer. Large Scale Global Optimization Using Differential Evolution with Self Adaptation and Cooperative Co-evolution. V *2008 IEEE World Congress on Computational Intelligence*, strani 3719–3726. IEEE Press, 2008.

[264] A. Zamuda, J. Brest, B. Bošković in V. Žumer. Študija samoprilagajanja krmilnih parametrov pri algoritmu DEMOwSA. *Elektrotehniški vestnik*, 75(4):223–228, 2008.

[265] A. Zamuda, J. Brest, B. Bošković in V. Žumer. Differential Evolution with Self-adaptation and Local Search for Constrained Multiobjective Optimization. V *IEEE Congress on Evolutionary Computation 2009*, strani 195–202. IEEE Press, 2009.

[266] A. Zamuda, J. Brest, B. Bošković in V. Žumer. Woody Plants Model Recognition by Differential Evolution. V *The Fourth International Conference on Bioinspired Optimization Methods and their Applications, May 20 - 21 2010, Ljubljana, Slovenia*, strani 205–215, 2010.

[267] A. Zamuda, J. Brest, B. Bošković in V. Žumer. Differential Evolution for Parameterized Procedural Woody Plant Models Reconstruction. *Applied Soft Computing*, 11:4904–4912, 2011.

[268] A. Zamuda, J. Brest, N. Guid in V. Žumer. Construction of Virtual Trees within Ecosystems with EcoMod Tool. V *Proceedings of IPSI-2006 Slovenia, International Conference on Advances in the Internet, Processing, Systems, and Interdisciplinary Research*, stran 15, 2006.

[269] A. Zamuda, J. Brest, N. Guid in V. Žumer. Modelling, Simulation, and Visualization of Forest Ecosystems. V *The IEEE Region 8 EUROCON 2007: International conference on "Computer as a tool", September 9-12, 2007, Warsaw, Poland*, strani 2600–2606. IEEE Press, 2007.

[270] A. Zamuda, J. Brest in V. Žumer. Razpoznava numerično kodiranih proceduralnih modelov iz slik naravnih dreves z uporabo diferencialne evolucije. V *Zbornik osemnajste mednarodne Elektrotehniške in računalniške konference ERK 2009, vol. B*, strani 171–174, 2009.

[271] A. Zamuda in D. Strnad. Interaktivni modelirnik realističnih animiranih dreves. *Zbornik trinajste mednarodne Elektrotehniške in računalniške konference ERK 2004*, strani 11–14, 2004.

[272] A. Zamuda, A. Čep in J. Brest. Optimizacija medatomskega energijskega potenciala Lennard-Jones z diferencialno evolucijo na arhitekturi CUDA. V *Zbornik dvajsete mednarodne Elektrotehniške in računalniške konference ERK 2011, vol. B*, strani 197–200, 2011.

[273] A. Zamuda. Modeliranje, simulacija in upodabljanje drevesnih ekosistemov. Diploma thesis, Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, jun 2006.

[274] J. Zhang in A. C. Sanderson. JADE: adaptive differential evolution with optional external archive. *Trans. Evol. Comp*, 13(5):945–958, 2009.

[275] J. Zhang in A. Sanderson. JADE: Self-adaptive differential evolution with fast and reliable convergence performance. *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, strani 2251–2258, 25-28 Sept. 2007.

[276] Q. Zhang in H. Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007.

[277] Q. Zhang in P. N. Suganthan. Final Report on CEC'09 MOEA Competition. Technical report, School of CS & EE, University of Essex, UK; School of EEE, Nanyang Technological University, Singapore, URL `http://web.mysites.ntu.edu.sg/epnsugan/PublicSite/Shared%20Documents/CEC2009-MOEA/Final-Compariosn-Report.pdf`, 2009.

[278] Q. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu in S. Tiwari. Multiobjective optimization Test Instances for the CEC 2009 Special Session and Competition. Technical Report CES-487, University of Essex, Colchester, UK and Nanyang Technological University, Singapore, Special Session on Performance Assessment of Multi-Objective Optimization Algorithms, Technical Report, 2008.

[279] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan in Q. Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1(1):32–49, 2011.

[280] K. Zielinski in R. Laur. Differential Evolution with Adaptive Parameter Setting for Multi-Objective Optimization. V *The 2007 IEEE Congress on Evolutionary Computation CEC 2007*, strani 3585–3592. IEEE Press, 2007.

[281] K. Zielinski, D. Peters in R. Laur. Run Time Analysis regarding Stopping Criteria for Differential Evolution and Particle Swarm Optimization. V *Proceedings of the 1st International Conference on Experiments/Process/System Modelling/Simulation/Optimization, Athens, Greece*. IC-EpsMsO, 2005.

[282] K. Zielinski, P. Weitkemper, R. Laur in K.-D. Kammeyer. Optimization of Power Allocation for Interference Cancellation With Particle Swarm Optimization. *IEEE Transactions on Evolutionary Computation*, 13(1):128–150, 2008.

[283] K. Zielinski in R. Laur. Adaptive Parameter Setting for a Multi-Objective Particle Swarm Optimization Algorithm. V *The 2007 IEEE Congress on Evolutionary Computation CEC 2007*, strani 3019–3026. IEEE Press, 2007.

[284] E. Zitzler, M. Laumanns in L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. V *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, uredniki: K. C. Giannakoglou, D. T. Tsahalis, J. Périaux, K. D. Papailiou in T. Fogarty, strani 95–100, Athens, Greece, 2001. International Center for Numerical Methods in Engineering (CIMNE).

[285] E. Zitzler in S. Künzli. Indicator-Based Selection in Multiobjective Search. V *Parallel Problem Solving from Nature (PPSN VIII)*, urednika: X. Yao et al., strani 832–842, Berlin, Germany, 2004. Springer-Verlag.

[286] E. Zitzler, M. Laumanns in L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, 2001.

Delovni življenjepis

Ime in priimek:	Aleš Zamuda	
Rojen:	23. 4. 1982	na Ptuju
Jeziki:		materni jezik slovenščina, tekoče angleščina in nemščina, osnove francoščine in hrvaščine
Šolanje:	1989 – 1997	Osnovna šola Velika Nedelja
	1997 – 2001	Gimnazija Ptuj
	2001 – 2012	Fakulteta za elektrotehniko, računalništvo in informatiko, Univerza v Mariboru (FERI UM)
Diploma:	2006	FERI UM: univerzitetni diplomirani inženir računalništva in informatike
Magisterij:	2008	FERI UM: magister znanosti (na področju računalništva in informatike)
Zaposlitev:	2006 –	Laboratorij za računalniške arhitekture in jezike, Fakulteta za elektrotehniko, računalništvo in informatiko, Univerza v Mariboru
Kontakt:	telefon	+386 2 220 7404
	e-pošta	ales.zamuda@uni-mb.si
	splet	http://www.aleszamuda.si/

Magister Aleš Zamuda se je rodil 23. 4. 1982 na Ptuju. Najprej se je šolal v Osnovni šoli Velika Nedelja od leta 1989. Strásti do programiranja je sledil že od prvega razreda osnovne šole. V času osnovnega šolanja je začel programirati v jezikih logo, basic, Pascal in C, spoznal jezike spletnih tehnologij in obiskal več privatnih računalniških šol na nacionalnem in mednarodnem nivoju. Šolanje je leta 1997 nadaljeval na gimnaziji Ptuj, ki jo je zaključil leta 2001. Leta 2006 je kot prvi diplomant generacije diplomiral na univerzitetnem študijskem programu računalništvo in informatika na Fakulteti za elektrotehniko, računalništvo in informatiko (FERI), Univerze v Mariboru (UM) ter se tam tudi zaposlil. Leta 2008 je na isti fakulteti s povprečno oceno 10,0 zagovoril magistrsko nalogo na področju računalništva in informatike. Ves čas srednjega in dodiplomskega visokošolskega šolanja je bil Zoisov štipendist. Leta 2008 je za raziskovalno delo prejel nagrado Andreja Perlacha UM. V času podiplomskega študija je prejel tudi tri nagrade FERI UM za raziskovalno delo in mednarodno nagrado IEEE.

Trenutno je zaposlen kot asistent z magisterijem na FERI UM. Je član Institute of Electrical and Electronics Engineers (IEEE) in recenzent pri številnih znanstvenih revijah z visokim faktorjem vpliva na računalniškem področju izmed katerih so nekatere navedene tudi v literaturi te disertacije. Skozi aktivno sodelovanjem na znanstvenih kongresih je vzpostavil stik tudi s številnimi mednarodnimi znanstveniki iz področij navedene literature in recenziral njihova dela s sodelovanjem v kongresnih programskih odborih. Njegove objave so deležne številnih citatov v znanstveni literaturi, sodeloval je tudi pri vodenju številnih diplom študentov na dodiplomskem študiju. Zmagal je na več nacionalnih in mednarodnih računalniških tekmovanjih, mednarodno objavil odprto-kodne programske pakete in prodal lastne računalniške aplikacije.

Delovne izkušnje

- Zaposlen na Fakulteti za računalništvo in informatiko v Laboratoriju za računalniške arhitekture in jezike, sprva kot tehniški sodelavec (2006 – 2009), nato kot asistent (od 2009),
  - delovno mesto asistent z magisterijem (2009), pedagoški naziv asistent (2006, 2009), raziskovalni naziv asistent (2007, 2009),
  - vodenje računalniških vaj pri predmetih: Računalniške arhitekture (2009/10, 2010/11, 2011/12), Heterogeni računalniški sistemi (2009/10, 2010/11, 2011/12), Povezljivi sistemi in inteligentne storitve (2010/11, 2011/12), Sistemska administracija (2010/11), Programiranje 1 UNI (2011/12)
  - vodenje študenstkih projektov Omrežne igre (2011–2012) in Mobilne igre (2012–2013)
  - član komisije za promocijo študija na FERI UM (od 2007)
  - predsednik centralne popisne komisije na FERI (2009)
- Zunanji sodelavec Fakultete za naravoslovje in matematiko UM, oddelek za matematiko in računalništvo
- član Institute of Electrical and Electronics Engineers (IEEE), član IEEE Computational Intelligence Society (CIS) (od 2006),
  - recenzent znanstvenih revij z visoko stopnjo citiranosti v znanosti (SCI): IEEE Transactions on Evolutionary Computation, IEEE transactions on systems, man and cybernetics. Part B. Cybernetics, Journal of industrial and management optimization, IEEE transactions on industrial electronics, International Journal of Systems Science (Taylor & Francis), Engineering Applications of Artificial Intelligence – Elsevier, Information Sciences – Elsevier,
  - član mednarodnih programskih komitejev na kongresih evolucijskega računanja IEEE (IEEE CEC 2011 (New Orleans, ZDA), IEEE SSCI 2011 (Pariz, Francija), IEEE SSCI 2012, IEEE WCCI 2010 (Barcelona, Španija), IEEE CEC 2009 (Trondheim, Norveška), IEEE WCCI 2008 (Hong Kong, Kitajska), IEEE CEC 2007 (Singapur) in drugih konferencah,
  - član Slovenian Artificial Intelligence Society (SLAIS) (od 2006),
  - sodelavec delavnic Algoritmi po vzoru iz narave (od 2006).
- z odliko opravljeno diplomsko delo v Laboratoriju za računalniško grafiko in umetno inteligenco z naslovom Modeliranje, simulacija in upodabljanje drevesnih ekosistemov, mentor red. prof. dr. Nikola Guid (2006),
- praksa v Centru za interdisciplinarne in multidisciplinarne raziskave in študije: delo na sistemu za e-izobraževanje Moodle v jeziku PHP (2006),
- praksa v Laboratoriju za računalniško grafiko in umetno inteligenco: izdelava simulatorja zaraščanja naravne pokrajine v jeziku C++ s Qt4 in OpenGL (2005),
- študentsko delo v Laboratoriju za heterogene računalniške sisteme: izdelava porazdeljenega sistema obdelave RSS novic v jeziku CSharp za občino Maribor (2005),
- dvakrat po enomesečna praksa v Ljubljani, v podjetju IxtlanTeam (2001, 2002),
- enotedenska praksa na Hermes Softlab (1998): grafični vmesniki in dinamično programiranje v Visual Basic ter vtičniška mrežna komunikacija,
- v domačem podjetju A&S, mrežno programiranje (v okolju Clipper) vodenja računovodskih kalkulacij za podjetje Kema (1995-1997).

Pedagoške delovne izkušnje

- Koordinator projektnega nabora predmetov Omrežne igre in Mobilne igre na ŠP R-IT FERI UM 1. bolonjske stopnje,
- komentor pri številnih diplomantih na 1. stopnji bolonjskega ŠP R-IT,
- asistent na univerzitetnih predmetih na 1. bolonjski stopnji ŠP R-IT FERI UM pri prof. Žumerju in prof. Brestu (Programiranje I UNI, Računalniške arhitekture, Sistemska administracija, Heterogeni računalniški sistemi) in 2. bolonjski stopnji ŠP R-IT FERI UM (Povezljivi sistemi in inteligentne storitve),
- tutor na visokošolskem študijskem programu R-IT,
- asistent na izrednem študijskem programu Elektrotehnika, predmet Računalništvo (2007),
- tehniški sodelavec v Laboratoriju za računalniške arhitekture in jezike FERI UM: Programiranje I, Računalništvo, Računalniške arhitekture, Principi programskih jezikov (2006),
- učitelj pri predmetu Algoritmi in programski jeziki za tečaj poklicnega izobraževanja (Srednja elektro-računalniška šola Maribor, 2008–2011),
- demonstrator pri predmetu Računalništvo na UM FERI, ŠP Elektrotehnika (2004/ 2005) in
- demonstrator pri računalniškem opismenjevanju na Ljudski univerzi Ormož (1997-2003).

Tehnološko znanje in izdelki

Tehnološko znanje: temeljito poznavanje in izkušnje s programskimi jeziki C, C++, C#, Java, Delphi, PHP, Ruby, Matlab, Python; temeljito poznavanje spletnih standardov HTML, CSS, XHTML, RSS, Web 2.0, ASP.NET; temeljito poznavanje aplikacijskih ogrodij Qt4, Win32 API, WinForms; *SQL; temeljito poznavanje razvojnih okolij QtCreator, KDevelop, MS Visual Studio, Eclipse, JDevelop in drugih; poznavanje tehnologij za paralelno procesiranje OpenMP, MPI, CUDA, GridGain; poznavanje zbirnega in strojnega jezika NASM; temeljito poznavanje hrambenih formatov pisarn OpenOffice.Org (Writer, Calc, Presentation), Microsoft Office (Word, Excel, PowerPoint); temeljito poznavanje jezika LaTeX in programiranje formata PDF; temeljito poznavanje konzolne in grafične uporabe ter programiranja gonilnikov za Linux in Windows; temeljito poznavanje sistemov BSD in Mac OS X, okolij KDE in GNOME; programiranje in vzpostavitev ogrodij CMS: MediaWiki, Joomla, Moodle; temeljito poznavanje programskih vmesnikov za OpenGL; večletne izkušnje programiranja umetne inteligence za svoje računalniške igre v 2D in 3D ter razvoj programskih jezikov; magisterij znanosti iz evolucijskih optimizacijskih algoritmov.

Izdelki: Evolucijski algoritem DEMOwSA, OpenGL animacijski paket EcoMod, Sistem za distribuirano zbiranje in obdelavo novic, spletišče Laboratorija za računalniške arhitekture in jezike, šahovski program QuTeChess, igra Štiri v vrsto (z nasprotnikovo UI), igra Šnops (z UI, voznik avtomobila), igra BlowUp, igra Kolo sreče.

Računalniška znanja pred pričetkom študija, samostojno učenje

- seminar programskega jezika java na UM FERI (1999),
- udeležba na mednarodnem kampu informatikov Hrvaške v Zadru (1997): dinamično programiranje v pascalu, OpenGL in BASH v Unixu,
- enotedenska poletna šola računalništva v Mariboru (1995): algoritmi v C++,
- enotedenska poletna šola programiranja v Bohinju podjetja Sinel (1994): grafično programiranje v pascalu,
- v osnovni šoli, samostojno učenje (samouk): programiranje svojih računalniških iger z 2D grafiko, zvokom in umetno inteligenco ter orodji za hiter razvoj aplikacij: v okolju Turbo Pascal 7.0 (prejetem kot nagrado za zmago na državnem tekmovanju iz znanja programiranja, 1994).

Mednarodna tekmovanja in nagrade

- ECiDUE na kongresu IEEE CEC 2009 (Trondheim, Norveška): 1. mesto (2. avtor), LSGO na kongresu IEEE WCCI 2008 (Hong Kong, Kitajska): 6. mesto (1. avtor) in MOEA na kongresu CEC 2007 (Singapur): 3. mesto, (1. avtor),
- Tekmovanje študentskih člankov IEEE Regije 8 (Evropa, Afrika, Bližnji vzhod) na konferenci EUROCON 2007 (Varšava, Poljska): 2. mesto (1. avtor).

Nacionalna tekmovanja in nagrade

- nagrada Univerze v Mariboru za raziskovalno delo (nagrada Andreja Perlacha) – ob Rektorjevem dnevu Univerze v Mariboru (2008) in tri priznanja FERI UM za znanstveno raziskovalno delo (2006, 2007, 2008),
- prvo mesto na študentskem tekmovanju Slovenske sekcije IEEE, Portorož (2006), spominsko darilo ministrice za visoko šolstvo, znanost in tehnologijo za vidne rezultate na mednarodnih tekmovanjih iz računalništva, ob sprejemu za študente, vodstva univerz in fakultet ter javnih raziskovalnih zavodov (2007), ter prva nagrada in zlata plaketa na natečaju za najbolj inventivni in podjetniško usmerjeni projekt Eureka! mladi (2008),
- prva nagrada iz znanja programiranja na srednješolskem državnem prvenstvu (2001), dvakrat osvojen naslov državnega prvaka v znanju iz programiranja (1995, 1997), naslov državnega prvaka iz oblikovanja besedil v programu Word (1997) ter osvojeno 5. mesto na tekmovanju iz programa CorelDraw (1997), ter dve zlati odličji v znanju iz kemije (1996, 1997) in srebrna nagrada za znanja iz matematike (1997),
- vpis v zlato knjigo Osnovne šole Velika Nedelja (1997), sprejem kot nadarjen šolar pri županu Občine Ormož (1997) in kot nadarjen dijak pri županu Občine Ptuj (2001) ter tvorno sodelovanje pri izdelavi spletnega mesta, ki je osvojilo 1. mesto na tekmovanju med osnovnimi in srednjimi šolami (1996).

Citiranost v znanstveni literaturi

Skupno število vseh citatov (15. maj 2012): Scopus – 80 (h-indeks: 5), Google Scholar – 201.
Citiranost različnih: avtorjev (20+), držav (11) in univerz (19) (vir: ResearcherID.com).
48 citatov: BREST, Janez, ZAMUDA, Aleš, BOŠKOVIĆ, Borko, SEPESY MAUČEC, Mirjam, ŽUMER, Viljem. High-dimensional real-parameter optimization using self-adaptive differential evolution algorithm with population size reduction. V: IEEE World Congress on Computational Intelligence, Hong Kong, June 1-6, 2008. *WCCI 2008 Proceedings*. [S. l.]: IEEE, 2008, str. 2032-2039.
38 citatov: ZAMUDA, Aleš, BREST, Janez, BOŠKOVIĆ, Borko, ŽUMER, Viljem. Large scale global optimization using differential evolution with self-adaptation and cooperative co-evolution. V: IEEE World Congress on Computational Intelligence, Hong Kong, June 1-6, 2008. *WCCI 2008 Proceedings*. [S. l.]: IEEE, 2008, str. 3719-3726.
34 citatov: ZAMUDA, Aleš, BREST, Janez, BOŠKOVIĆ, Borko, ŽUMER, Viljem. Differential evolution for multiobjective optimization with self adaptation. V: *CEC 2007: 2007 Congress on Evolutionary Computation: 25-28 September 2007, Singapore*. Piscataway: IEEE, 2007, str. 3617-3624.
30 citatov: BREST, Janez, ZAMUDA, Aleš, BOŠKOVIĆ, Borko, SEPESY MAUČEC, Mirjam, ŽUMER, Viljem. Dynamic optimization using self-adaptive differential evolution. V: *2009 IEEE Congress on Evolutionary Computation, IEEE CEC 2009, 18th to 21st May 2009, Trondheim, Norway: programme & abstracts*. Piscataway: IEEE, 2009, str. 415-422.
11 citatov: BREST, Janez, ZAMUDA, Aleš, BOŠKOVIĆ, Borko, GREINER, Sašo, ŽUMER, Viljem. An analysis of the control parameters' adaptation in DE. V: *Advances in differential evolution*, (Studies in computational intelligence, Vol. 143). Berlin; Heidelberg: Springer, 2008, str. 89-110.
10 citatov: ZAMUDA, Aleš, BREST, Janez, BOŠKOVIĆ, Borko, ŽUMER, Viljem. Differential evolution with self-adaptation and local search for constrained multiobjective optimization. V: *CEC 2009: 2009 Congress on Evolutionary Computation, Trondheim, Norveška*. Piscataway: IEEE, 2009, str. 195-202.

Osebna bibliografija

ČLANKI IN DRUGI SESTAVNI DELI

1.01 Izvirni znanstveni članek

1. ZAMUDA, Aleš, BREST, Janez, BOŠKOVIĆ, Borko, ŽUMER, Viljem. Študija samoprilagajanja krmilnih parametrov pri algoritmu DEMOwSA. *Elektrotehniški vestnik*. [Slovenska tiskana izd.], 2008, letn. 75, št. 4, str. 223-228. [COBISS.SI-ID 12933654]

2. BREST, Janez, ZAMUDA, Aleš, BOŠKOVIĆ, Borko, ŽUMER, Viljem. Globalna optimizacija problemov z velikim številom dimenzij. *Elektrotehniški vestnik*. [Slovenska tiskana izd.], 2008, letn. 75, št. 5, str. 299-304. [COBISS.SI-ID 13038614]

3. ZAMUDA, Aleš, BREST, Janez, BOŠKOVIĆ, Borko, ŽUMER, Viljem. Differential evolution for parameterized procedural woody plant models reconstruction. *Applied soft computing*, 2011, letn. 11, št. 8, str. 4904-4912, doi: 10.1016/j.asoc.2011.06. 009. [COBISS.SI-ID 15175446]

4. BREST, Janez, KOROŠEC, Peter, ŠILC, Jurij, ZAMUDA, Aleš, BOŠKOVIĆ, Borko, SEPESY MAUČEC, Mirjam. Differential evolution and differential ant-stigmergy on dynamic optimisation problems. *Int. J. Syst. Sci.*, Dostopno online: 26 sep. 2011, doi: 10.1080/00207721. 2011.617899.[COBISS.SI-ID 15354390]

5. BOŠKOVIĆ, Borko, BREST, Janez, ZAMUDA, Aleš, GREINER, Sašo, ŽUMER, Viljem. History mechanism supported differential evolution for chess evaluation function tuning. *Soft computing*. [Tiskana izd.], 2011, letn. 15, št. 4, str. 667-683, doi: 10.1007/s00500-010-0593-z. [COBISS.SI-ID 13985046]

6. ZAMUDA, Aleš, BREST, Janez. Population reduction differential evolution with multiple mutation strategies in real world industry challenges. *Lect. notes comput. sci.*, str. 154-161, ilustr. [COBISS.SI-ID 15979542] tipologija 1.08 → 1.01

1.08 Objavljeni znanstveni prispevek na konferenci

7. ZAMUDA, Aleš, STRNAD, Damjan. Interaktivni modelirnik realističnih animiranih dreves. V: ZAJC, Baldomir (ur.). *Zbornik trinajste mednarodne elektrotehniške in računalniške konference ERK 2004, 27. - 29. september 2004, Portorož, Slovenija*. Ljubljana: IEEE Region 8, Slovenska sekcija IEEE, 2004, zv. B, str. 11-14. [COBISS.SI-ID 9058326]

8. ZAMUDA, Aleš, BREST, Janez, GUID, Nikola, ŽUMER, Viljem. Construction of virtual trees within ecosystems with ecomod tool. V: International Conference on Advances in the Internet, Processing, Systems and Interdisciplinary Research, Bled, November 30-December 3, 2006. *Proceedings of the VIPSI-2006 Slovenia: Multidisciplinary, Interdisciplinary, Transdisciplinary: M+I+T++, Bled Slovenia, November 30-December 3, 2006*. Belgrade: IPSI, 2006, [5] f. [COBISS.SI-ID 11005206]

9. ZAMUDA, Aleš, BREST, Janez, BOŠKOVIĆ, Borko, ŽUMER, Viljem. Differential evolution for multiobjective optimization with self adaptation. V: *CEC 2007: 2007 Congress on Evolutionary Computation: 25-28 September 2007, Singapore*. Piscataway: Institute of Electrical and Electronics Engineers, 2007, str. 3617-3624. [COBISS.SI-ID 11730710]

10. ZAMUDA, Aleš, BREST, Janez, GUID, Nikola, ŽUMER, Viljem. Modelling, simulation and visualisation of forest ecosystems. V: The International Conference on Computer as a tool, Warsaw, Poland, September 9-12, 2007. *Region 8 Eurocon 2007: proceedings*. Piscataway, NJ: IEEE Service Center, [2007], str. 2600-2606. [COBISS.SI-ID 11643926]

11. BOŠKOVIĆ, Borko, BREST, Janez, ZAMUDA, Aleš, ŽUMER, Viljem. Uglaševanje šahovskega programa BBChess z uporabo algoritma diferencialne evolucije. V: ZAJC, Baldomir (ur.), TROST, Andrej (ur.). *Zbornik Šestnajste mednarodne Elektrotehniške in računalniške konference ERK 2007, 24.-26. september 2007, Portorož, Slovenija*. Ljubljana: IEEE Region 8, Slovenska sekcija IEEE, 2007, zv. B, str. 73-76. [COBISS.SI-ID 11760662]

12. BREST, Janez, ZAMUDA, Aleš, BOŠKOVIĆ, Borko, ŽUMER, Viljem. Večkriterijska optimizacija: primerjava algoritmov v MOjDE in DEMO. V: ZAJC, Baldomir (ur.), TROST, Andrej (ur.). *Zbornik Šestnajste mednarodne Elektrotehniške in računalniške konference ERK 2007, 24. -26. september 2007, Portorož, Slovenija*. Ljubljana: IEEE Region 8, Slovenska sekcija IEEE, 2007, zv. B, str. 85-88. [COBISS.SI-ID 11773718]

13. ZAMUDA, Aleš, BREST, Janez, BOŠKOVIĆ, Borko, ŽUMER, Viljem. Večkriterijska optimizacija: eksperimentalni rezultati algoritmov MOjDE in DEMO. V: ZAJC, Baldomir (ur.), TROST, Andrej (ur.). *Zbornik Šestnajste mednarodne Elektrotehniške in računalniške konference ERK 2007, 24. -26. september 2007, Portorož, Slovenija*. Ljubljana: IEEE Region 8, Slovenska sekcija IEEE, 2007, zv. B, str. 89-92. [COBISS.SI-ID 11773974]

14. BREST, Janez, ZAMUDA, Aleš, BOŠKOVIĆ, Borko, GREINER, Sašo, SEPESY MAUČEC, Mirjam, ŽUMER, Viljem. Self-adaptive differential evolution with SQP local search. V: FILIPIČ, Bogdan (ur.), ŠILC, Jurij (ur.). *Bioinspired optimization methods and their applications: proceedings of the Third International Conference on Bioinspired Optimization Methods and their Applications, BIOMA 2008, 13-14 October 2008, Ljubljana, Slovenia*. Ljubljana: Jožef Stefan Institute, 2008, str. 59-69. [COBISS.SI-ID 12707862]

15. BREST, Janez, ZAMUDA, Aleš, BOŠKOVIĆ, Borko, SEPESY MAUČEC, Mirjam, ŽUMER, Viljem. High-dimensional real-parameter optimization using self-adaptive differential evolution algorithm with population size reduction. V: IEEE World Congress on Computational Intelligence, Hong Kong, June 1-6, 2008. *WCCI 2008 Proceedings*. [S. l.]: IEEE, 2008, str. 2032-2039. [COBISS.SI-ID 12328982]

16. ZAMUDA, Aleš, BREST, Janez, BOŠKOVIĆ, Borko, ŽUMER, Viljem. Large scale global optimization using differential evolution with self-adaptation and cooperative co-evolution. V: IEEE World Congress on Computational Intelligence, Hong Kong, June 1-6, 2008. *WCCI 2008 Proceedings*. [S. l.]: IEEE, 2008, str. 3719-3726. [COBISS.SI-ID 12329238]

17. ZAMUDA, Aleš, BREST, Janez, BOŠKOVIĆ, Borko, ŽUMER, Viljem. Diferencialna evolucija za večkriterijsko optimizacijo s samoprilagajanjem in z lokalnim preiskovanjem SQP. V: ZAJC, Baldomir (ur.), TROST, Andrej (ur.). *Zbornik sedemnajste mednarodne Elektrotehniške in računalniške konference ERK 2008, 29. september - 1. oktober 2008, Portorož, Slovenija*, (Zbornik ... Elektrotehniške in računalniške konference ERK ...). Ljubljana: IEEE Region 8, Slovenska sekcija IEEE, 2008, zv. B, str. 103-106. [COBISS.SI-ID 12695574]

18. BOŠKOVIĆ, Borko, BREST, Janez, ZAMUDA, Aleš, ŽUMER, Viljem. Ratingiranje pri uglaševanju šahovskega programa z algoritmom diferencialne evolucije. V: ZAJC, Baldomir (ur.), TROST, Andrej (ur.). *Zbornik sedemnajste mednarodne Elektrotehniške in računalniške konference ERK 2008, 29. september - 1. oktober 2008, Portorož, Slovenija*, (Zbornik ... Elektrotehniške in računalniške konference ERK ...). Ljubljana: IEEE Region 8, Slovenska sekcija IEEE, 2008, zv. B, str. 123-126. [COBISS.SI-ID 12695830]

19. ZAMUDA, Aleš, BREST, Janez, BOŠKOVIĆ, Borko, ŽUMER, Viljem. Differential evolution with self-adaptation and local search for constrained multiobjective optimization. V: *2009 IEEE Congress on Evolutionary Computation, IEEE CEC 2009, 18th to 21st May 2009, Trondheim, Norway: programme & abstracts*. Piscataway: IEEE, 2009, str. 195-202. [COBISS.SI-ID 13217558]

20. BREST, Janez, ZAMUDA, Aleš, BOŠKOVIĆ, Borko, SEPESY MAUČEC, Mirjam, ŽUMER, Viljem. Dynamic optimization using self-adaptive differential evolution. V: *2009 IEEE Congress on Evolutionary Computation, IEEE CEC 2009, 18th to 21st May 2009, Trondheim, Norway: programme & abstracts*. Piscataway: IEEE, 2009, str. 415-422. [COBISS.SI-ID 13217814]

21. ZAMUDA, Aleš, BREST, Janez, ŽUMER, Viljem. Razpoznava numerično kodiranih proceduralnih modelov iz slik naravnih dreves z uporabo diferencialne evolucije. V: ZAJC, Baldomir (ur.), TROST, Andrej (ur.). *Zbornik Osemnajste mednarodne elektrotehniške in računalniške konference - ERK 2009, 21-23. september 2009, Portorož, Slovenija*. Ljubljana: IEEE Region 8, Slovenska sekcija IEEE, 2009, zv. B, str. 171-174, ilustr. [COBISS.SI-ID 13463318]

22. ZAMUDA, Aleš, BREST, Janez, BOŠKOVIĆ, Borko, ŽUMER, Viljem. Woody plants model recognition by differential evolution. V: FILIPIČ, Bogdan (ur.), ŠILC, Jurij (ur.). *Bioinspired optimization methods and their applications: proceedings of the Fourth International Conference on Bioinspired Optimization Methods and their Applications, BIOMA 2010, 20-21 May 2010, Ljubljana, Slovenia*. Ljubljana: Jožef Stefan Institute, 2010, str. 205-215. [COBISS.SI-ID 14126102]

23. BREST, Janez, ZAMUDA, Aleš, FISTER, Iztok, SEPESY MAUČEC, Mirjam. Large scale global optimization using self-adaptive differential evolution algorithm. V: IEEE World Congress on Computational Intelligence, Barcelona, July 18-23, 2010. *WCCI 2010 Proceedings*. [S. l.]: IEEE, cop. 2010, CEC IEEE, str. 3097-3104. [COBISS.SI-ID 14281238]

24. ZAMUDA, Aleš, BREST, Janez. Večkriterijska rekonstrukcija numerično kodiranih proceduralnih modelov dreves z diferencialno evolucijo. V: ZAJC, Baldomir (ur.), TROST, Andrej (ur.). *Zbornik devetnajste mednarodne Elektrotehniške in računalniške konference ERK 2010, Portorož, Slovenija, 20.-22. september 2010*, (Zbornik ... Elektrotehniške in računalniške konference ERK ...). Ljubljana: IEEE Region 8, Slovenska sekcija IEEE, 2010, zv. B, str. 169-172. [COBISS.SI-ID 14408726]

25. BREST, Janez, ZAMUDA, Aleš, FISTER, Iztok, BOŠKOVIĆ, Borko, SEPESY MAUČEC, Mirjam. Constrained real-parameter optimization using a differential evolution algorithm. V: *IEEE SSCI2011 symposium series on computational intelligence: proceedings*. Piscataway: IEEE, cop. 2011, str. 9-16. [COBISS.SI-ID 14952214]

26. ZAMUDA, Aleš. Diferencialna evolucija realnih industrijskih izzivov CEC 2011 = Differential evolution of RealWorld Industry Challenges CEC 2011. V: ZAJC, Baldomir (ur.), TROST, Andrej (ur.). *Zbornik dvajsete mednarodne Elektrotehniške in računalniške konference ERK 2011, 19. - 21. september 2011, Portorož, Slovenija*, (Zbornik ... Elektrotehniške in računalniške konference ERK ...). Ljubljana: IEEE Region 8, Slovenska sekcija IEEE, 2011, zv. B, str. 185-188. [COBISS.SI-ID 15361302]

27. ZAMUDA, Aleš, ČEP, Aleš, BREST, Janez. Optimizacija medatomskega energijskega potenciala Lennard-Jones z diferencialno evolucijo na arhitekturi CUDA = Interatomic Lennard-Jones potential energy optimization using differential evolution on CUDA architecture. V: ZAJC, Baldomir (ur.), TROST, Andrej (ur.). *Zbornik dvajsete mednarodne Elektrotehniške in računalniške konference ERK 2011, 19. - 21. september 2011, Portorož, Slovenija*, (Zbornik ... Elektrotehniške in računalniške konference ERK ...). Ljubljana: IEEE Region 8, Slovenska sekcija IEEE, 2011, zv. B, str. 197-200. [COBISS.SI-ID 15362582]

1.12 Objavljeni povzetek znanstvenega prispevka na konferenci

28. ZAMUDA, Aleš, GUID, Nikola. Modeliranje, simulacija in upodabljanje gozdov. V: ZAJC, Baldomir (ur.), TROST, Andrej (ur.). *Zbornik petnajste mednarodne Elektrotehniške in računalniške konference ERK 2006, 25. - 27. september 2006, Portorož, Slovenija*, (Zbornik ... Elektrotehniške in računalniške konference ERK ...). Ljubljana: IEEE Region 8, Slovenska sekcija IEEE, 2006, zv. B, str. 391-392. [COBISS.SI-ID 10776598]

29. ZAMUDA, Aleš. Testne funkcije in metrike za večkriterijsko optimzacijo. V: FILIPIČ, Bogdan (ur.), MERNIK, Marjan (ur.), PAPA, Gregor (ur.). *Zbornik povzetkov delavnic "Algoritmi po vzorih iz narave" v študijskem letu 2006/2007: Ljubljana, september 2007*. Ljubljana: Institut "Jožef Stefan", 2007, str. 13. [COBISS.SI-ID 12763670]

30. ZAMUDA, Aleš. Večkriterijska optimizacija: eksperimentalni rezultati algoritmov MOjDE in DEMO. V: FILIPIČ, Bogdan (ur.), MERNIK, Marjan (ur.), PAPA, Gregor (ur.). *Zbornik povzetkov delavnic "Algoritmi po vzorih iz narave" v študijskem letu 2006/2007: Ljubljana, september 2007*. Ljubljana: Institut "Jožef Stefan", 2007, str. 28. [COBISS.SI-ID 12768534]

31. ZAMUDA, Aleš. Interaktivno gojenje ekosistemov z operatorji evolucije. V: FILIPIČ, Bogdan (ur.), MERNIK, Marjan (ur.), PAPA, Gregor (ur.). *Zbornik povzetkov delavnic "Algoritmi po vzorih iz narave" v študijskem letu 2007/2008: Ljubljana, september 2008*. Ljubljana: Institut "Jožef Stefan", 2008, str. 11. [COBISS.SI-ID 12755478]

32. ZAMUDA, Aleš. Diferencialna evolucija z večkriterijsko optimizacijo s samoprilagajanjem in z lokalnim preiskovanjem SQP. V: FILIPIČ, Bogdan (ur.), MERNIK, Marjan (ur.), PAPA, Gregor (ur.). *Zbornik povzetkov delavnic "Algoritmi po vzorih iz narave" v študijskem letu 2007/2008: Ljubljana, september 2008*. Ljubljana: Institut "Jožef Stefan", 2008, str. 22. [COBISS.SI-ID 12756502]

1.16 Samostojni znanstveni sestavek ali poglavje v monografski publikaciji

33. BREST, Janez, ZAMUDA, Aleš, BOŠKOVIĆ, Borko, GREINER, Sašo, ŽUMER, Viljem. An analysis of the control parameters' adaptation in DE. V: *Advances in differential evolution*, (Studies in computational intelligence, Vol. 143). Berlin; Heidelberg: Springer, 2008, str. 89-110. [COBISS.SI-ID 12503830]

34. BOŠKOVIĆ, Borko, GREINER, Sašo, BREST, Janez, ZAMUDA, Aleš, ŽUMER, Viljem. An adaptive differential evolution algorithm with opposition-based mechanisms, applied to the tuning of a chess program. V: *Advances in differential evolution*, (Studies in computational intelligence, Vol. 143). Berlin; Heidelberg: Springer, 2008, str. 287-298. [COBISS.SI-ID 12504854]

MONOGRAFIJE IN DRUGA ZAKLJUČENA DELA

2.09 Magistrsko delo

35. ZAMUDA, Aleš. *Samoprilagajanje krmilnih parametrov pri algoritmu diferencialne evolucije za večkriterijsko optimizacijo: magistrsko delo*. Maribor: [A. Zamuda], 2008. XVI, 112 str., graf. prikazi, tabele. http://dkum.uni-mb.si/Dokument.php?id=6502. [COBISS.SI-ID 12461846]

2.11 Diplomsko delo

36. ZAMUDA, Aleš. *Modeliranje, simulacija in upodabljanje drevesnih ekosistemov: diplomska naloga univerzitetnega študijskega programa*, (Fakulteta za elektrotehniko, računalništvo in informatiko, Diplomska dela univerzitetnega študja). Maribor: [A. Zamuda], 2006. 98 f., ilustr. [COBISS.SI-ID 10822166]

2.12 Končno poročilo o rezultatih raziskav

37. ZAZULA, Damjan, ŽUMER, Viljem, GUID, Nikola, ŽALIK, Borut, OJSTERŠEK, Milan, RIZMAN ŽALIK, Krista, MERNIK, Marjan, PODGORELEC, David, POTOČNIK, Božidar, BREST, Janez, KOLMANIČ, Simon, LENIČ, Mitja, STRNAD, Damjan, CIGALE, Boris, GOMBOŠI, Matej, REBERNAK, Damijan, HOLOBAR, Aleš, KRIVOGRAD, Sebastian, DIVJAK, Matjaž, MUNDA, Jurij, KLAJNŠEK, Gregor, ZADRAVEC, Mirko, HERIC, Dušan, KOSAR, Tomaž, DOMITER, Vid, HORVAT, Marjan, TUTEK, Simon, NERAT, Andrej, ŠPELIČ, Denis, PIVEC, Boštjan, BREZOVNIK, Janez, ISTENIČ, Rok, OBRUL, Denis, ZAMUDA, Aleš. *Računalniški sistemi, metodologije in inteligentne storitve: zaključno poročilo o rezultatih raziskovalnega programa v obdobju 2004-2008*. Maribor: Fakulteta za elektrotehniko, računalništvo in informatiko, 2009. 1 mapa (loč. pag.). [COBISS.SI-ID 14079766]

38. ZAZULA, Damjan, GUID, Nikola, ŽALIK, Borut, OJSTERŠEK, Milan, MERNIK, Marjan, BREST, Janez, HOLOBAR, Aleš, PODGORELEC, David, KOLMANIČ, Simon, KOSAR, Tomaž, ZAMUDA, Aleš, BOŠKOVIĆ, Borko, FISTER, Iztok. *Računalniški sistemi, metodologije in inteligentne storitve: letno poročilo o rezultatih raziskovalnega programa v letu 2009: programska skupina P2-0041*. Maribor: Fakulteta za elektrotehniko, računalništvo in informatiko, 2010. 1 mapa (loč. pag.). [COBISS.SI-ID 14078486]

2.21 Programska oprema

39. ZAMUDA, Aleš, STRNAD, Damjan, BREST, Janez, BOŠKOVIĆ, Borko. *Drevo*. [s. l.: s. n., 2007]. 1 CD ROM. `http://labraj.uni-mb.si/~ales/codes/drevo.zip`. [COBISS.SI-ID 11607062]

40. BOŠKOVIĆ, Borko, BREST, Janez, ZAMUDA, Aleš. *Šahovski program BBChess 1.1*. Maribor: [s. n., 2007]. 1 CD ROM. `http://labraj.uni-mb.si/images/6/6e/BBChess1.1_src.tar.bz`. [COBISS.SI-ID 11873046]

41. ZAMUDA, Aleš, BREST, Janez. *FastFractal "DoubleDip" C++2Java Interface*. Maribor: [s. n., 2008]. 1 CD ROM. http://www.cs.bham.ac.uk/research/projects/ecb/data/150/cec08-f7-cpp-with-sources-1.0RC1.zip, http://nical.ustc.edu.cn/conferences/cec2008/lsgo/LSGO.CEC08 .Benchmark.zip. [COBISS.SI-ID 12207894]

42. ZAMUDA, Aleš. *DECMOSA-SQP*. Maribor: [s. n., 2009]. 1 CD ROM. `http://web.mysites.ntu.edu.sg/epnsugan/PublicSite/Shared%20Documents/CEC2009-MOEA/Codes/raw-unchanged-source-of-DECMOSA-SQP-CEC09.tar`, `http://labraj.uni-mb.si/~ales/codes/raw-unchanged-source-of-DECMOSA-SQP-CEC09.tgz`. [COBISS.SI-ID 14134550]

IZVEDENA DELA (DOGODKI)

3.14 Predavanje na tuji univerzi

43. ZAMUDA, Aleš. *Heterogeneous computer system in the view of ubiquitous computing and connectible intelligent services: lecture on the Universidad de Las Palmas de Gran Canaria, Canary Islands, Spain, EU, 7-11 May, 2012 (ERASMUS teaching staff mobillity programme)*. 2012. [COBISS.SI-ID 15993110]

3.25 Druga izvedena dela

44. BOŠKOVIĆ, Borko, GREINER, Sašo, ZAMUDA, Aleš. *Tečaj Linux: tečaj za Dravske elektrarne Maribor (DEM) in Holding slovenskih elektrarn (HSE), Maribor, 4.-12. junij 2009*. Maribor: Fakulteta za elektrotehniko, računalništvo in informatiko, Laboratorij za računalniške arhitekture in jezike, 2009. [COBISS.SI-ID 13264662]

45. ZAMUDA, Aleš. *Evolucijska rekonstrukcija drevesnih modelov: razvojno-raziskovalni prispevek Laboratorija za računalniške arhitekture in jezike na Delavnici programske skupine Računalniški sistemi, metodologije in inteligentne storitve, Fakuteta za elektrotehniko, računalništvo in informatiko Maribor, 10. februar 2011*. Maribor, 2011. [COBISS.SI-ID 15296534]

SEKUNDARNO AVTORSTVO

Komentor pri diplomskih delih (bolonjski študij 1. stopnje)

46. BEZGET, Jan. *Iskanje in vizualizacija poti na morju: diplomsko delo visokošolskega strokovnega študijskega programa*. Maribor: [J. Bezget], 2010. VII, 43 str., ilustr. `http://dkum.uni-mb.si/Dokument.php?id=17525`. [COBISS.SI-ID 14733334]

47. OSTOJIĆ, Danijel. *Klici oddaljenih spletnih storitev in spletni založniški sistem: diplomsko delo univerzitetnega študijskega programa*. Maribor: [D. Ostojić], 2010. X, 48 str., ilustr. `http://dkum.uni-mb.si/Dokument.php?id=17420`. [COBISS.SI-ID 14758166]

48. ŠTRUC, Nace. *Paralelno programiranje s pomočjo standarda OpenMP: diplomska naloga univerzitetnega študijskega programa*. Maribor: [N. Štruc], 2010. 32 str., ilustr. `http://dkum.uni-mb.si/Dokument.php?id=17207`. [COBISS.SI-ID 14511126]

49. JURŠIČ, Janko. *Problem simetričnega trgovskega potnika in optimizacija z genetskimi algoritmi: diplomsko delo univerzitetnega študijskega programa*. Maribor: [J. Juršič], 2010. VI, 26 f., ilustr. `http://dkum.uni-mb.si/Dokument.php?id=18399`. [COBISS.SI-ID 14782742]

50. PAJNIK, Aleš. *Spletni založniški sistem "JOOMLA!": diplomsko delo univerzitetnega študijskega programa*. Maribor: [A. Pajnik], 2010. IX, 34 str., ilustr. `http://dkum.uni-mb.si/Dokument.php?id=17960`. [COBISS.SI-ID 14772502]

51. REBSELJ, Boris. *Vizualizacija zgradb s programskima vmesnikoma Google Maps in Google Earth: diplomsko delo visokošolskega strokovnega študijskega programa*. Maribor: [B. Rebselj], 2010. VIII, 52 f., ilustr. `http://dkum.uni-mb.si/Dokument.php?id=19933`. [COBISS.SI-ID 14915350]

52. MLINAR, Tilen. *Aplikacija za prikaz sledenja vozil na operacijskem sistemu iOS : diplomsko delo univerzitetnega študijskega programa*. Maribor: [T. Mlinar], 2011. VIII, 27 f., ilustr. `http://dkum.uni-mb.si/Dokument.php?id=24719`. [COBISS.SI-ID 15650070]

53. KUKOVIČ, Denis. *Izdelava spletljaja za Kolab in Sharepoint 2010 : diplomsko delo univerzitetnega študijskega programa*. Maribor: [D. Kukovič], 2011. VII, 43 f., ilustr. `http://dkum.uni-mb.si/Dokument.php?id=26163`. [COBISS.SI-ID 15872534]

54. ČEP, Aleš. *Optimizacija medatomskega potenciala Lennard-Jones s paralelno diferencialno evolucijo: diplomsko delo univerzitetnega študijskega programa*. Maribor: [A. Čep], 2011. IX, 27 f., ilustr. `http://dkum.uni-mb.si/Dokument.php?id=24817`. [COBISS.SI-ID 15353622]

55. BRENČIČ, Boštjan. *Optimizacija s čebelami: diplomsko delo univerzitetnega študijskega programa*. Maribor: [B. Brenčič], 2011. VII, 29 f., ilustr. `http://dkum.uni-mb.si/Dokument.php?id=25254`. [COBISS.SI-ID 15371286]

56. ROJKO, Lovro. *Umetna inteligenca za simulacijo vožnje avtomobila po dirkalni progi: diplomsko delo univerzitetnega študijskega programa*. Maribor: [L. Rojko], 2011. IX, 32 f., ilustr. `http://dkum.uni-mb.si/Dokument.php?id=24818`. [COBISS.SI-ID 15623702]

Pisec recenzij

57. *Engineering applications of artificial intelligence*. Zamuda, Aleš (pisec recenzij 2010, 2011). [Print ed.]. Swansea: Pineridge Press, 1988-. ISSN 0952-1976. [COBISS.SI-ID 25396224]

58. *IEEE transactions on evolutionary computation*. Zamuda, Aleš (pisec recenzij 2007, 2008, 2009, 2010). [Print ed.]. New York: Institute of Electrical and Electronics Engineers, 1997-. ISSN 1089-778X. [COBISS.SI-ID 16940805]

59. *IEEE transactions on industrial electronics*. Zamuda, Aleš (pisec recenzij 2009). [Print ed.]. New York: Institute of Electrical and Electronics Engineers, 1982-. ISSN 0278-0046. [COBISS.SI-ID 8726789]

60. *IEEE transactions on systems, man and cybernetics. Part B. Cybernetics*. Zamuda, Aleš (pisec recenzij 2009, 2010). [Print ed.]. New York (N.Y.): Institute of Electrical and Electronics Engineers, 1996-. ISSN 1083-4419. [COBISS.SI-ID 16640005]

61. *Informatica*. Zamuda, Aleš (pisec recenzij 2010). Ljubljana: Slovene Society Informatika, 1977-. ISSN 0350-5596. [COBISS.SI-ID 746244]

62. *Information sciences*. Zamuda, Aleš (pisec recenzij 2011). [Print ed.]. New York: North-Holland, 1968-. ISSN 0020-0255. [COBISS.SI-ID 25613056]

63. *International Journal of Systems Science*. Zamuda, Aleš (pisec recenzij 2010, 2011). London: Taylor & Francis, 1970-. ISSN 0020-7721. [COBISS.SI-ID 25657856]

64. *Journal of industrial and management optimization*. Zamuda, Aleš (pisec recenzij 2008). Springfield, Mo.: American Institute of Mathematical Sciences, 2005-. ISSN 1547-5816. [COBISS.SI-ID 12967190]

65. *Memetic computing*. Zamuda, Aleš (pisec recenzij 2012). Heidelberg; Berlin: Springer. ISSN 1865-9292. `http://www.springerlink.com/content/121309`. [COBISS.SI-ID 518445081]

Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko

IZJAVA DOKTORSKEGA KANDIDATA

Podpisani **mag. Aleš Zamuda**,

vpisna številka **95034129**

i z j a v l j a m,

da je doktorska disertacija z naslovom **Diferencialna evolucija za rekonstrukcijo parametriziranih proceduralnih drevesnih modelov**

- rezultat lastnega raziskovalnega dela,
- da predložena disertacija v celoti ali v delih ni bila predložena za pridobitev kakršnekoli izobrazbe po študijskem programu druge fakultete ali univerze,
- da so rezultati korektno navedeni in
- da nisem kršil avtorskih pravic in intelektualne lastnine drugih.

Podpis doktorskega kandidata:

Aleš Zamuda

Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko

IZJAVA KANDIDATOVEGA MENTORJA O USTREZNOSTI DOKTORSKE DISERTACIJE

Podpisani **red. prof. dr. Janez Brest**, mentor doktorskemu kandidatu, izjavljam, da je doktorska disertacija z naslovom

Diferencialna evolucija za rekonstrukcijo parametriziranih proceduralnih drevesnih modelov,

ki jo je izdelal doktorski kandidat **mag. Aleš Zamuda**,

v skladu z odobreno temo, Pravilnikom o pripravi in zagovoru doktorske disertacije ter mojimi navodili in predstavlja izviren prispevek k razvoju znanstvene discipline.

Datum in kraj:

Maribor, 15. 5. 2012

Podpis mentorja:

UNIVERZA V MARIBORU

Fakulteta za elektrotehniko, računalništvo in informatiko

IZJAVA O ISTOVETNOSTI TISKANE IN ELEKTRONSKE VERZIJE ZAKLJUČNEGA DELA IN OBJAVI OSEBNIH PODATKOV DOKTORANDOV

Ime in priimek doktoranda-dke: Aleš Zamuda

Vpisna številka: 95034129

Študijski program: FERI-RI DR RAČUNALNIŠTVO IN INFORMATIKA DR

Naslov doktorskega dela: Diferencialna evolucija za rekonstrukcijo parametriziranih proceduralnih drevesnih modelov

Mentor: Janez Brest

Somentor:

Podpisani-a Aleš Zamuda **izjavljam, da sem za potrebe arhiviranja oddal elektronsko verzijo zaključnega dela v Digitalno knjižnico Univerze v Mariboru. Doktorsko delo sem izdelal-a sam-a ob pomoči mentorja. V skladu s 1. odstavkom 21. člena Zakona o avtorskih in sorodnih pravicah dovoljujem, da se zgoraj navedeno zaključno delo objavi na portalu Digitalne knjižnice Univerze v Mariboru.**

Tiskana verzija doktorskega dela je istovetna elektronski verziji, ki sem jo oddal za objavo v Digitalno knjižnico Univerze v Mariboru.

Zaključno delo zaradi zagotavljanja konkurenčne prednosti, varstva industrijske lastnine ali tajnosti podatkov naročnika:
ne sme biti javno dostopno do _____ **(datum odloga javne objave ne sme biti daljši kot 3 leta od zagovora dela).**

Podpisani izjavljam, da dovoljujem objavo osebnih podatkov vezanih na zaključek študija (ime, priimek, leto in kraj rojstva, datum doktoriranja, naslov doktorskega dela) na spletnih straneh in v publikacijah UM.

Datum in kraj: Maribor, 18.05.2012

Podpis doktoranda-dke: Aleš Zamuda

Podpis mentorja _____
(samo v primeru, če delo ne sme biti javno dostopno):

Podpis odgovorne osebe naročnika in žig: _____
(samo v primeru, če delo ne sme biti javno dostopno)