UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Erik Dovgan

EVOLUCIJSKI ALGORITEM ZA OPTIMIRANJE PREVOZA TOVORA MED DVEMA LOKACIJAMA S SKUPINO VOZIL

Diplomska naloga
na univerzitetnem študiju

Mentor: doc. dr. Janez Demšar
Somentor: doc. dr. Bogdan Filipič

Ljubljana, 2008

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Namesto te strani **vstavite** original izdane teme diplomskega dela s podpisom mentorja in dekana ter žigom fakultete, ki ga diplomant dvigne v študentskem referatu, preden odda izdelek v vezavo!

Zahvala

Mentorju doc. dr. Janezu Demšarju in somentorju doc. dr. Bogdanu Filipiču se zahvaljujem za vodenje, usmerjanje in pomoč pri izdelavi diplomske naloge. Posebna zahvala gre družini in prijateljem, ki so me vselej podpirali, ter mi po potrebi tudi pomagali in svetovali.

Moji družini in prijateljem

Kazalo

Povzetek

Naloga obravnava uporabo evolucijskega algoritma za optimiranje prevoza tovora med dvema lokacijama s skupino vozil. Z naraščanjem velikosti problema njegova računska zahtevnost narašča tako, da postane prezahteven, da bi ga reševali z determinističnimi algoritmi. Zato smo razvili evolucijski algoritem, ki spada med stohastične algoritme. Te algoritme uporabljamo pri reševanju težkih problemov, katerih čas reševanja z determinističnimi algoritmi je nesprejemljiv. Pomanjkljivost stohastičnih algoritmov pa je, da velikokrat ne najdejo optimalne rešitve. Večinoma najdejo suboptimalno rešitev oziroma lokalni optimum ocenitvene funkcije.

V diplomi je opisan problem optimiranja prevoza tovora med dvema lokacijama s skupino vozil. Opisane so lastnosti vozil, ki jih uporabljamo za prevoz tovora, lastnosti nakladalnih mest ter lastnosti razkladalnih mest. Opisane so tudi splošne lastnosti evolucijskih algoritmov, njihovi mehanizmi delovanja ter vrste evolucijskih algoritmov, ki so genetski algoritmi, evolucijske strategije, evolucijsko programiranje, genetsko programiranje in diferencialna evolucija. V nadaljevanju je opisan razviti evolucijski algoritem, ki je bil testiran z vnaprej določenimi vrednostmi parametrov na štirih testnih problemih. Rezultate smo primerjali z rezultati, dobljenimi s požrešno metodo. Pri večini problemov evolucijski algoritem vrne boljše rezultate od požrešne metode. Opisana je tudi uglasitev evolucijskega algoritma z metaevolucijskim algoritmom. Predstavljeni so rezultati uglašenega evolucijskega algoritma, njihova primerjava z rezultati požrešne metode in možni načini izboljšave algoritma.

Ključne besede:

prevoz tovora, vožnja v koloni, optimiranje prevoza, evolucijski algoritem, uglaševanje parametrov, metaevolucijski algoritem, požrešna metoda

Abstract

This thesis deals with the optimization of heterogeneous cargo transport between two locations with a group of vehicles. With the increasing size the problem becomes too complex to be solved with deterministic algorithms. Therefore we designed an evolutionary algorithm which belongs to the family of stochastic algorithms. This kind of algorithms is used for solving problems whose solving time with deterministic algorithms would be unacceptable. A disadvantage of stochastic algorithms is that they frequently do not find the optimal solution. Usually they find a suboptimal solution which is a local optimum of the evaluation function.

In this thesis we describe the optimization of heterogeneous cargo transport between two locations with a group of vehicles. We begin by formally defining the problem in terms of cargo and vehicle characteristics. Then we present the evolutionary algorithm characteristics and their examples: genetic algorithms, evolutionary strategies, evolutionary programming, genetic programming and differential evolution. Next we describe the evolutionary algorithm implemented to solve the presented problem which was tested with the predefined parameter values on four test problems. The results were compared with those of the greedy algorithm. The evolutionary algorithm mostly found better results than the greedy algorithm. The algorithm parameters were tuned with a metaevolutionary algorithm which is also described. At the end we present the results obtained with the metaevolutionary algorithm, their comparison with the results of the greedy algorithm and the future work.

Keywords:

cargo transport, group transport, transport optimization, evolutionary algorithm, parameter tuning, metaevolutionary algorithm, greedy algorithm

Poglavje 1

Uvod

Evolucijske algoritme uvrščamo med stohastične algoritme. Ti algoritmi so se izkazali za učinkovito metodo optimizacije, preiskovanja in strojnega učenja na mnogih realnih problemih. Klasične deterministične optimizacijske postopke je težko uporabiti za reševanje problemov z visoko zahtevnostjo, saj čas reševanja zahtevnih problemov s klasičnimi postopki presega za uporabnike sprejemljiv čas reševanja. Za reševanje takih problemov potrebujemo stohastične algoritme. Ti algoritmi imajo manjšo časovno zahtevnost v primerjavi z determinističnimi metodami. Njihov problem pa je, da večinoma ne najdejo optimalne rešitve. Ponavadi najdejo suboptimalno rešitev oziroma lokalni optimum ocenitvene funkcije rešitev [7].

Problem optimiranja prevoza tovora je zelo zahteven zaradi velikega števila možnih rešitev. S povečevanjem velikosti problema njegova računska zahtevnost narašča tako, da postane prezahteven, da bi ga reševali z determinističnimi algoritmi. Zato smo za reševanje problema uporabili evolucijski algoritem.

Problem optimiranja prevoza tovora med dvema lokacijama s skupino vozil je opisan v drugem poglavju. Predstavljene so definicije in omejitve v problemu ter cilj reševanja problema. V tretjem poglavju so predstavljeni evolucijski algoritmi ter njihovo delovanje. Poleg tega je predstavljenih tudi več vrst evolucijskih algoritmov: genetski algoritmi, evolucijske strategije, evolucijsko programiranje, genetsko programiranje in diferencialna evolucija. Evolucijski algoritem za reševanje obravnavanega problema je opisan v četrtem poglavju. V petem poglavju so predstavljeni primeri problemov ter rezultati reševanja teh problemov. Opisana je tudi uglasitev algoritma z metaevolucijskim algoritmom in primerjava rešitev, ki jih vrne evolucijski algoritem, z rešitvami, ki jih vrne požrešna metoda.

Poglavje 2

Opis problema

Potrebno je prepeljati tovor iz skladišča v kraju A v skladišče v kraju B v čim krajšem času. Imamo v različnih vrst tovora. Tovor vozimo z vozili z različnimi lastnostmi. Prevoz se izvaja v koloni.

Za prevoz je na voljo n vozil. Za vsako vozilo je znana nosilnost vsake izmed vrst tovora, ki je označena z w_{ij}, kjer sta $i = 1, \ldots, n$ in $j = 1, \ldots, v$. Nosilnost w_{ij} je podana v isti enoti kot količina tovora. Primeri enot so kilogrami, kubični metri, kosi itd. Pri tem ni nujno, da vozilo lahko vozi vse vrste tovora. Če vozilo i ne more voziti vrste tovora j, potem je $w_{ij} = 0$.

Vozila lahko vozijo več vrst tovora. Kljub temu prevoz poteka tako, da vsako vozilo naenkrat vozi le eno vrsto tovora. Vozilo pa lahko v posameznih vožnjah vozi različne vrste tovora. Poleg tega se na vsako vozilo vedno naloži toliko tovora, kolikor je njegova nosilnost. Izjema so le vozila, za katera ob nakladanju določene vrste tovora zmanjka te vrste tovora.

Pot iz kraja A v kraj B, po kateri vozijo vozila, je natančno definirana. Razdeljena je na cestne odseke, ki se razlikujejo predvsem po različnih kategorijah cest. Za vsak odsek je podana dolžina in največja dovoljena hitrost. Za vsako vozilo je predpisana največja hitrost vožnje, ko je polno naloženo, ter največja hitrost vožnje, ko je prazno. Iz teh podatkov se za vsako vozilo i izračunata časa vožnje med krajema A in B, ko je vozilo polno, $t_i^{(1)}$, in ko je vozilo prazno, $t_i^{(0)}$. Za vožnjo iz kraja A v kraj B se vedno vzame čas polnega vozila, tudi če ni polno naloženo. Za vožnjo iz kraja B v kraj A pa se vedno vzame čas praznega vozila.

V kraju A je na voljo m nakladalnih mest. Vsako nakladalno mesto ima podano hitrost nakladanja vsake izmed vrst tovora. Hitrosti nakladalnih mest so označene z a_{ij}, kjer je $i = 1, \ldots, m$ in $j = 1, \ldots, v$. Vsaka izmed hitrosti je definirana kot količina vrste tovora v ustrezni enoti, ki jo lahko na nakla-

dalnem mestu naložimo v eni uri. Če na nakladalnem mestu i ne moremo nakladati vrste tovora j, potem je $a_{ij} = 0$. V kraju B je k razkladalnih mest, ki imajo, enako kot nakladalna mesta, podane hitrosti razkladanja vrst tovora, ki so označene z b_{ij}, kjer je $i = 1, \ldots, k$ in $j = 1, \ldots, v$. Za b_{ij} veljajo enake značilnosti kot za a_{ij}.

V določenem trenutku se lahko na nakladalnem oziroma razkladalnem mestu nahaja le eno vozilo. Nakladanje in razkladanje potekata, dokler vozilo ni polno oziroma dokler ne zmanjka tovora, kar pomeni, da nakladanje in razkladanje potekata brez prekinitev.

Za vsako vrsto tovora i, kjer je $i = 1, \ldots, v$, je podana količina tovora V_i, ki ga je potrebno prepeljati iz kraja A v kraj B. Ker je ponavadi količina tovora tolikšna, da je ni mogoče prepeljati v eni vožnji, je potrebno opraviti več voženj oziroma ciklov. Če pa je količina tovora tolikšna, da jo vozila prepeljajo v enem ciklu, potem optimiramo samo ta cikel.

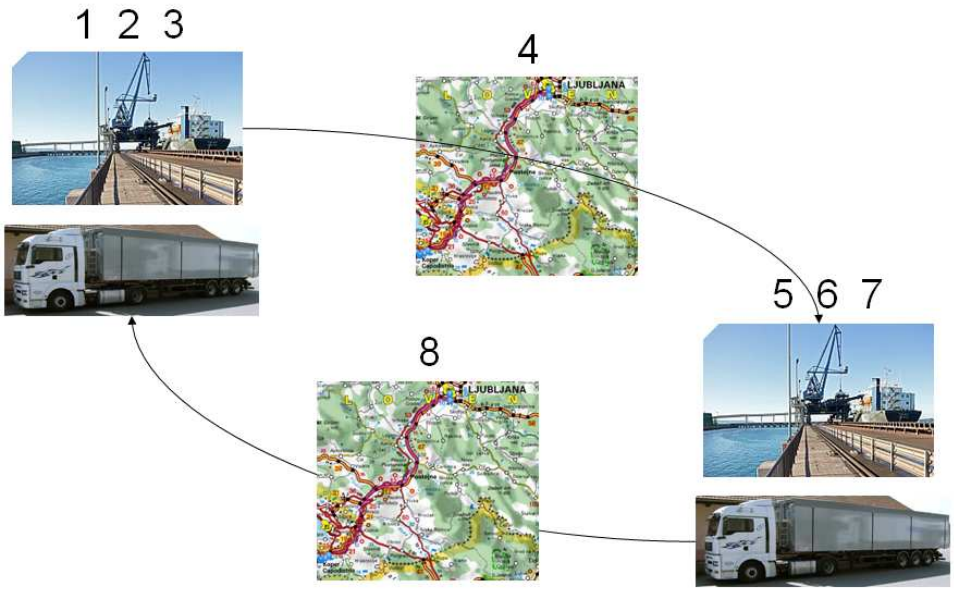
Prevoz vsega tovora se tako izvede v enem ali več ciklih. Vsak cikel se začne in konča v kraju A. Sestavljen je iz naslednjih korakov:

1. čakanje vozil pred dodeljenimi nakladalnimi mesti, da se ta sprostijo,
2. nakladanje dodeljenih vrst tovora na nakladalnih mestih,
3. čakanje, da je naloženo zadnje izmed vozil na enem od nakladalnih mest,
4. vožnja v koloni iz kraja A v kraj B,
5. čakanje vozil pred dodeljenimi razkladalnimi mesti, da se ta sprostijo,
6. razkladanje tovora na razkladalnih mestih,
7. čakanje, da je zadnje izmed vozil razloženo na enem od razkladalnih mest,
8. vožnja v koloni iz kraja B v kraj A.

Koraki cikla so prikazani na sliki 2.1.

Prevoz v koloni pomeni, da je potrebno v kraju A počakati, da se vsa vozila naložijo, da pri prevozu iz kraja A v kraj B kolona porabi toliko časa, kolikor ga porabi najpočasnejše polno vozilo, da je potrebno v kraju B počakati, da se vsa vozila razložijo, in da pri prevozu iz kraja B v kraj A kolona porabi toliko časa, kolikor ga porabi najpočasnejše prazno vozilo.

Vozila so lahko opremljena s samonakladalnimi napravami. Samonakladalne naprave lahko nakladajo in razkladajo tovor. Ta lastnost vozil je pomembna predvsem v primeru, ko ni na razpolago ustreznih mest za nakladanje



Slika 2.1: Koraki cikla.

oziroma razkladanje tovora. Poleg tega takim vozilom ni potrebno čakati v vrsti, da se nakladalno oziroma razkladalno mesto sprosti. Opremljenost vozila s samonakladalno napravo pa zmanjša nosilnost tega vozila, saj samonakladalna naprava zasede določen prostor, na katerega bi se lahko naložil tovor. Zato v primeru, ko so na voljo vozila s samonakladalnimi napravami ter vozila brez samonakladalnih naprav, ni nujno boljša izbira vozil s samonakladalnimi napravami. V primeru, ko se večino časa porabi za prevoz med krajema, manj časa pa za nakladanje in razkladanje, je bolj smiselno uporabiti vozila brez samonakladalnih naprav, saj imajo večjo zmogljivost in se z njimi v enem ciklu prepelje več tovora. Ko pa se večino časa porabi za nakladanje oziroma razkladanje, manj časa pa za prevoz med krajema, je bolj smiselno uporabiti vozila s samonakladalnimi napravami, ki hitreje nakladajo in razkladajo tovor in s tem pospešijo tisti del cikla, za katerega se porabi največ časa.

V primeru uporabe samonakladalne naprave predpostavljamo, da je vozilu omogočen takojšen dostop do tovora, ki ga je potrebno naložiti, oziroma do prostora, kamor je potrebno razložiti tovor. To pomeni, da tako vozilo ne čaka na samonakladanje oziroma samorazkladanje, kot morajo čakati vozila na nakladalnih oziroma razkladalnih mestih. Kljub tej prednosti pa opremljenost vozila s samonakladalno napravo ne pomeni nujne uporabe te naprave za samonakladanje oziroma samorazkladanje. Če je potrebno prepeljati še dve vrsti

tovora, ki ju lahko dve vozili v celoti prepeljata v enem ciklu, in če sta na voljo dve nakladalni mesti, se lahko izkaže, da bi samonakladanje enega izmed teh dveh vozil zahtevalo daljši čas, če je hitrost nakladanja nakladalnega mesta večja od hitrosti samonakladalne naprave. V večini primerov pa se uporaba samonakladalne naprave izkaže kot smiselna, saj poleg tega, da se vozilo takoj naloži oziroma razloži, to vozilo ne zaseda nakladalnega oziroma razkladalnega mesta in s tem pospeši nakladanje oziroma razkladanje kolone.

Če je vozilo i opremljeno s samonakladalno napravo, potem ima ta naprava podani hitrost samonakladanja a_{ij}^S in hitrost samorazkladanja b_{ij}^S za vsako vrsto tovora $j = 1, \ldots, v$. Če naprava ne more nakladati oziroma razkladati vrste tovora j, potem je $a_{ij}^S = 0$ oziroma $b_{ij}^S = 0$.

Pri razporejanju vozil je potrebno za vsako vozilo v vsakem ciklu določiti, katero vrsto tovora bo vozilo, na katerem nakladalnem mestu se bo nakladalo ali če bo uporabilo samonakladalno napravo, ter na katerem razkladalnem mestu se bo razkladalo ali če bo uporabilo samonakladalno napravo.

Cilj je poiskati tak razpored vrst tovora na vozila ter vozil na nakladalna in razkladalna mesta, da bo ves tovor prepeljan v čim krajšem času. Prevoz se konča, ko je ves tovor prepeljan iz kraja A v kraj B in se vsa vozila vrnejo v kraj A.

Poglavje 3

Evolucijski algoritmi

3.1 Splošno o evolucijskih algoritmih

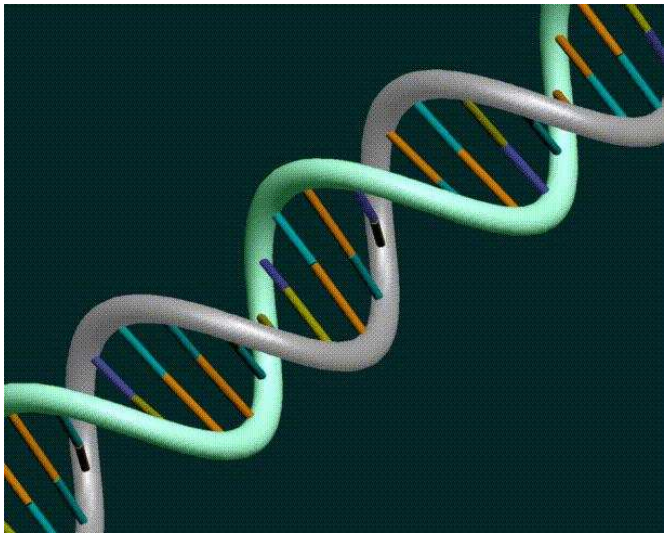
3.1.1 Primerjava z realnim svetom

Osnovna zakonitost evolucije je, da se lastnosti živih organizmov prenašajo iz generacije v generacijo [9]. Primeri lastnosti, ki se prenašajo pri ljudeh, so barva oči, barva las itd. Te lastnosti so določene z geni. Geni so zapisani na kromosomih v obliki deoksiribonukleinske kisline (DNK). Nahajajo se v celičnem jedru, po prepisu na ribonukleinsko kislino (RNK) pa predstavljajo navodila za tvorbo beljakovin. Tvorba beljakovin je osnova življenja. Vsaka celica ima specifično nalogo, ki je določena z DNK.

DNK je sestavljena iz dveh povezanih spiral, kot je prikazano na sliki 3.1. Vsako izmed spiral sestavljajo organske baze: A (adenin), T (timin), C (citozin) in G (gvanin). Te tvorijo le dva bazna para A-T in C-G, kot je prikazano na sliki 3.2. Če se na določenem položaju prve spirale nahaja baza T, se lahko na povezanem položaju druge spirale nahaja le A. Posledično lahko ob poznavanju ene spirale popolnoma rekonstruiramo drugo.

Nove celice nastajajo z delitvijo, ki jo imenujemo mitoza. Preden se celica razdeli, se mora DNK podvojiti, kot je prikazano na sliki 3.2. Obe novi celici imata tako enake gene kot materinska celica. Pri tem procesu včasih pride tudi do napake oziroma mutacije. Takšen gen imenujemo mutiran gen in lahko povzroči nastanek drugačnih beljakovin ali prepreči njihovo tvorbo.

Delitev spolnih celic imenujemo mejoza. Spolne celice imajo pri človeku le polovico kromosomov običajne celice. Pred delitvijo spolne celice se kromosomi temeljito premešajo, nato pa povežejo v pare. Potem se razdelijo na več delov in sestavijo nazaj skupaj tako, da se vsak drugi del zamenja med



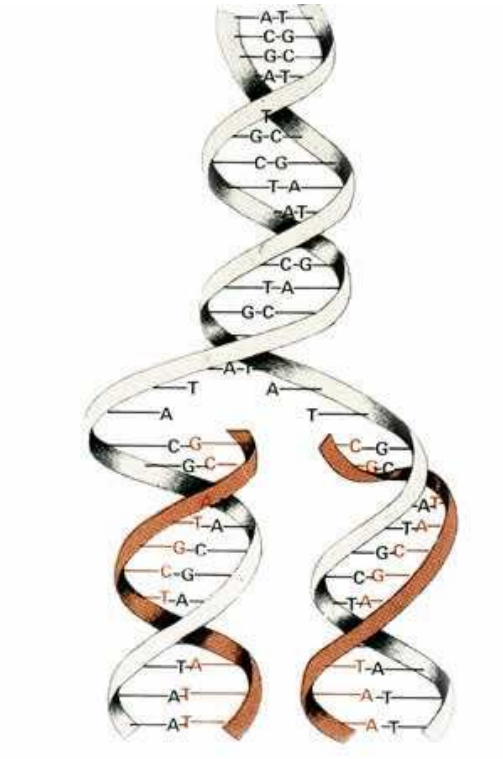
Slika 3.1: Dve povezani spirali deoksiribonukleinske kisline.

kromosomoma. Ta proces imenujemo križanje.

Te stalne spremembe so osnova za postopno spreminjanje organizmov. Če je spremenjeni organizem neustrezen glede na svoje okolje, ta organizem umre. Če pa se spremenjeni organizem okolju prilagaja bolje od prvotnega, potem lahko ta organizem izpodrine iz okolja nespremenjene organizme. Ta proces imenujemo naravna selekcija. V okolju lahko obstaja le končna množica organizmov. Ker se organizmi množijo med seboj, je naravna selekcija osnovni mehanizem ohranjanja omejene množice organizmov in onemogočanja eksponentne rasti njihovega števila.

Funkcija ocenjevanja prilagoditve organizma okolju ima ponavadi več lokalnih optimumov. Pri razmnoževanju organizmov lahko pride do odstopajočih pojavov. Tako se lahko organizem z najboljšo oceno ne razmnoži in se njegov DNK izgubi. Poleg tega lahko z razmnoževanjem in naravno selekcijo preživijo organizmi, katerih ocena konvergira proti lokalnemu optimumu ocenitvene funkcije. S pomočjo mutacije in križanja lahko nastanejo organizmi, katerih ocena se nahaja na področju globalnega optimuma ocenitvene funkcije. Tako se populacija izogne zaustavitvi razvoja v lokalnem optimumu.

Mutacija in križanje nista dovolj za nastanek nove vrste. Nova vrsta nastane, ko se skupine osebkov določene vrste več ne množijo med seboj zaradi geografskih ovir ali pa zaradi mutacije. Po več generacijah, v katerih prihaja do mutacij in križanj, se ti dve skupini že toliko razlikujeta, da sta to že dve ločeni vrsti [10].



Slika 3.2: Podvojitev deoksiribonukleinske kisline.

3.1.2 Delovanje evolucijskih algoritmov

Razvoj algoritmov za reševanje problemov je ena izmed osrednjih tematik matematične in računalniške znanosti. Pri tem lahko poskušamo oponašati načine reševanja problemov, ki jih najdemo v naravi. Dva najmočnejša mehanizma reševanja problemov v naravi sta:

- človeški možgani, ki so ustvarili kolo, vojno, mesta itd.,
- evolucijski proces, ki je ustvaril možgane.

Algoritmi, ki poskušajo oponašati delovanje človeških možganov, se imenujejo nevronske mreže, algoritmi, ki poskušajo oponašati delovanje evolucijskega procesa, pa se imenujejo evolucijski algoritmi [1].

Zahtevnost problemov, s katerimi se soočamo v sodobnem svetu, narašča. Zmogljivost računalnikov se povečuje, a ne dovolj, da bi lahko rešili čedalje zahtevnejše probleme. Zato potrebujemo zmogljive algoritme, ki lahko v sprejemljivem času rešujejo široko množico problemov brez prekomernega prila-

gajanja posamičnemu problemu. Evolucijski algoritmi v veliki meri ustrezajo tem zahtevam.

Osnovna ideja evolucijskih algoritmov je računalniška simulacija evolucije osebkov, ki predstavljajo možne rešitve danega problema. Imamo populacijo osebkov in okolje, v katerem se osebki nahajajo. Okolje osebke pri razmnoževanju omejuje. Mehanizem omejevanja se imenuje naravna selekcija in povzroči, da se osebki z nižjo oceno prilagojenosti ne množijo. Tako se povprečna ocena osebkov v populaciji skozi generacije veča. Pri razmnoževanju osebkov se izvedeta križanje in mutacija. Križanje poteka med dvema ali več osebki in kot rezultat dobimo enega ali več novih osebkov. Mutacija se izvede nad enim osebkom in kot rezultat dobimo en nov osebek. Novi osebki tekmujejo s starimi osebki za mesto v populaciji nove generacije, pri čem se uporabi ocena na podlagi prilagoditve okolju ter včasih tudi na podlagi starosti. Proces se ciklično ponavlja, dokler se ne doseže zahtevane kakovosti osebkov oziroma dokler ne poteče omejitev računanja [7].

Pri tem algoritmu sta bistvena mehanizma:

- spreminjanje osebkov, ki obsega križanje in mutacijo,
- naravna selekcija, ki omogoča izboljševanje kakovosti osebkov.

Veliko komponent evolucijskega algoritma je stohastičnih. Boljši osebki imajo večjo verjetnost razmnoževanja, a tudi slabši osebki se lahko razmnožujejo ali ostanejo v populaciji, le verjetnost tega je manjša. Pri križanju so mesta, med katerimi se zamenjajo geni, izbrana naključno. Tudi pri mutaciji je gen, ki se mutira, izbran naključno. Poleg tega so tudi nove, mutirane vrednosti izbrane naključno [4].

Evolucijski algoritem lahko zapišemo v psevdokodi na naslednji način:

```
ZAČETEK
    naključno inicializiraj populacijo
    oceni vsak osebek
    PONAVLJAJ, DOKLER (ustavitveni pogoj ni izpolnjen)
        izberi starša
        križaj starša
        mutiraj osebka
        oceni osebka
```

izberi osebka za populacijo nove generacije

KONEC PONAVLJANJA

KONEC

V nadaljevanju predstavljamo najpomembnejše elemente evolucijskega algoritma: predstavitev rešitev, vrednotenje rešitev, selekcija, genetski operatorji in ustavitveni pogoj.

3.1.3 Predstavitev rešitev

Prvi korak pri implementaciji evolucijskega algoritma je predstavitev možnih rešitev obravnavanega problema. Vsaka možna rešitev je predstavljena kot niz genov, imenovan kromosom. Če so na primer rešitve optimizacijskega problema cela števila, potem jih lahko predstavimo v dvojiški obliki. V tem primeru so geni dvojiška števila. Lahko pa so geni cela števila. Zapisu rešitev v kromosomih pravimo kodiranje rešitev.

3.1.4 Vrednotenje rešitev

Za vrednotenje rešitev je potrebno definirati ocenitveno funkcijo, ki ji pravimo tudi funkcija uspešnosti. Funkcija se uporablja pri selekciji in pri tem omogoča izboljševanje populacije. Oblikujemo jo na podlagi cilja optimizacijskega problema, katerega rešujemo. Tipično je ta funkcija kvantitativna ocena kromosoma, cilj pa maksimizacija ali minimizacija te ocene. Če optimiramo funkcijo realnega sveta, potem je ocenitvena funkcija identična funkciji realnega sveta ali pa njena transformacija.

3.1.5 Selekcija

Selekcija osebkov, ki se bodo množili, temelji na izboru boljših osebkov glede na njihovo oceno. Pri tem zagotavlja, da imajo boljši osebki večjo verjetnost razmnoževanja, slabši pa manjšo, a vedno večjo od 0, saj bi v nasprotnem primeru iskanje postalo preveč požrešno in bi prehitro zašlo v lokalni optimum. Selekcija skupaj z izborom osebkov, ki se bodo uvrstili v naslednjo generacijo, zagotavlja, da se kakovost populacije povečuje.

3.1.6 Genetski operatorji

Genetski operatorji omogočajo nastajanje novih osebkov oziroma rešitev iz starih osebkov oziroma rešitev. Pravimo, da tvorijo nove osebke [8]. Najpogosteje uporabljana genetska operatorja sta križanje in mutacija.

Križanje

Križanje je operator, ki deluje nad dvema osebkoma. Ta operator združi informacije iz dveh osebkov v enega ali dva nova osebka. Križanje je stohastični operator. Naključno so izbrani deli osebkov, ki se zamenjajo. Križanje več kot dveh osebkov je mogoče, a nima ekvivalenta v naravi in ga zato redko uporabljamo v algoritmih.

Ideja križanja je, da ob združevanju dobrih lastnosti dveh osebkov dobimo boljši osebek. Ob križanju ni nujno, da dobimo dopustne rešitve. Poleg tega velikokrat ne pride do izboljšave. Kljub temu, da je verjetnost izboljšave majhna in da je potrebno veliko parov osebkov križati, da dobimo boljše nove osebke, pa križanje izvajamo z določeno verjetnostjo, tako da ne križamo vseh osebkov v populaciji.

Mutacija

Mutacija je operator, ki deluje nad enim osebkom. Pri tem nastane malo spremenjen nov osebek. Podobno kot križanje je tudi mutacija stohastični operator. Naključno se izbere gen kromosoma in naključno se mu določi nova vrednost.

Mutacija omogoča izhod osebka iz lokalnega optimuma in skok proti globalnemu optimumu, ker lahko naključno izbrana vrednost gena povzroči zelo veliko spremembo osebka. Poleg tega preprečuje nepovratne izgube, do katerih na primer pride, če imajo vsi osebki populacije enako vrednost določenega gena. Nekatere implementacije evolucijskih algoritmov omejujejo mutacijo na manjše spremembe osebkov in tako ne omogočajo izhoda iz lokalnega optimuma.

3.1.7 Ustavitveni pogoj

V evolucijskih algoritmih lahko ustavitveni pogoj definiramo na več načinov. Če poznamo optimalno rešitev, lahko ustavimo algoritem, ko najde optimalno rešitev, ponavadi v mejah določene natančnosti.

Ker je evolucijski algoritem stohastični algoritem, obstaja možnost, da algoritem ne najde optimalne rešitve. Zato definiramo tudi enega izmed naslednjih ustavitvenih pogojev:

1. največji čas izvajanja algoritma,
2. največje število ocenitev osebkov,
3. v določenem času, po določenem številu ocenitev osebkov ali po določenem številu generacij se ocena najboljšega osebka ne spremeni,
4. raznolikost populacije pade pod določeno mejo.

Včasih se uporabi disjunktna kombinacija več pogojev.

3.2 Vrste evolucijskih algoritmov

Poznamo več vrst evolucijskih algoritmov. Predstavili bomo naslednje algoritme [7]:

- genetski algoritmi,
- evolucijske strategije,
- evolucijsko programiranje,
- genetsko programiranje,
- diferencialna evolucija.

3.2.1 Genetski algoritmi

Genetski algoritmi so najbolj znana vrsta evolucijskih algoritmov. Obstaja več vrst genetskih algoritmov.

Primer genetskega algoritma je enostavni, kanonični genetski algoritem. Pri enostavnem genetskem algoritmu rešitve predstavimo z dvojiškimi števili, križanje je enomestno, mutacijo pa izvedemo s spremembo vrednosti izbranega bita. Verjetnost mutacije je zelo nizka [8].

Pri genetskih algoritmih obstaja več vrst predstavitev rešitev, na primer:

- dvojiška predstavitev,
- celoštevilska predstavitev,

- predstavitev z realnimi števili,
- predstavitev s permutacijami.

Verjetnost mutacije je zelo nizka. Pri dvojiški predstavitvi ob mutaciji spremenimo vrednost bita. Ob celoštevilski predstavitvi lahko mutacija dodeli genu naključno celoštevilsko vrednost ali pa samo prišteje ali odšteje določeno (manjše) celo število. Podobno genu z realno vrednostjo dodelimo naključno vrednost, ali pa mu prištejemo ali odštejemo določeno (manjše) realno število. Pri predstavitvi s permutacijami pa ni možna mutacija enega samega gena, saj se lahko določena vrednost pojavi v permutaciji le enkrat. Mutacija zato spremeni več genov naenkrat.

Nasprotno od mutacije pa je verjetnost križanja zelo visoka, tipično višja od 0,5. Križanje se ponavadi izvede nad dvema kromosomoma – staršema, možna pa je tudi izvedba nad več starši. Najenostavnejše je enomestno križanje, ki pa ga lahko razširimo na večmestno križanje. Poleg izmenjave genov med mesti križanja obstaja tudi vrsta križanja, imenovana uniformno križanje, kjer za vsak gen z neko verjetnostjo določimo, ali bo ta gen del prvega ali drugega potomca. Križanje več kot dveh staršev ni pogosta operacija v genetskih algoritmih. Učinkovitost take operacije je odvisna od obravnavanega problema. Kljub temu se je izkazala kot uporabna v veliko primerih.

Kromosome, ki jih bomo križali in mutirali, izberemo s selekcijo. Po izvedbi križanja in mutacije določimo, kateri izmed staršev in potomcev bodo ostali v populaciji. Glede na določitev osebkov, ki ostanejo v populaciji, ločimo dva modela genetskih algoritmov:

- generacijski model,
- model s stabilnim stanjem.

Pri generacijskem modelu celotno populacijo zamenjamo s potomci. Ob tem izgubimo vse visoko ocenjene kromosome iz populacije prejšnje generacije, kar je slabo. Nasprotno pa pri modelu s stabilnim stanjem le starše z nižjo oceno od potomcev zamenjamo s potomci. Možno je tudi, da potomci zamenjajo najslabše osebke v populaciji. To pa ponavadi vodi k prehitri konvergenci proti lokalnemu optimumu.

3.2.2 Evolucijske strategije

Evolucijske strategije (ES) se razlikujejo od ostalih evolucijskih algoritmov po tem, da se določeni parametri algoritma spreminjajo sočasno s potekom optimizacije in sicer tako, da so ti parametri vključeni v kromosome [10].

Evolucijske strategije tipično uporabljamo za optimizacijo zveznih spremenljivk. Prevladujoč evolucijski operator je mutacija. Parametri mutacije se spreminjajo med izvajanjem algoritma in so shranjeni v kromosomih.

Mutacija spreminja vrednosti spremenljivk glede na Gaussovo (normalno) porazdelitev, pri kateri je povprečna vrednost 0 in standardna deviacija parameter mutacije, ki se spreminja, optimira tekom izvajanja algoritma. Sprememba se zgodi tako, da izberemo naključno število na dani Gaussovi krivulji, ki ga prištejemo vrednosti izbranega gena. Večja je verjetnost izbora števil, ki so bliže ničli.

Pri križanju iz dveh staršev tvorimo potomca, katerega geni so lahko povprečje vrednosti istoležnih genov staršev ali pa naključno izbrani izmed istoležnih genov staršev.

Starše izberemo naključno. Evolucijske strategije ne poznajo drugih mehanizmov izbora staršev, kot je na primer selekcija na osnovi uspešnosti.

Samoprilagajanje parametrov je najpomembnejša značilnost evolucijskih strategij. Spreminjamo standardno deviacijo, ki je na začetku iskanja tipično velika in omogoča velike spremembe genov, s potekom optimiranja pa jo manjšamo, saj kromosomi skonvergirajo k optimumu in večje spremembe niso več potrebne, treba je le najti optimum.

Poznamo dva načina izbire kromosomov, ki sestavljajo populacijo naslednje generacije. Pri $(\mu+\lambda)$-ES izmed μ staršev in λ potomcev izberemo μ najboljših kromosomov, ki sestavljajo populacijo naslednje generacije. Pri (μ, λ)-ES, kjer je $\lambda > \mu$, izmed λ potomcev izberemo μ najboljših kromosomov, ki sestavljajo populacijo naslednje generacije. Pri tem načinu zavržemo trenutno najboljše starševske kromosome, a se tako izognemo lokalnemu optimumu in sledimo globalnemu optimumu [8].

3.2.3 Evolucijsko programiranje

Evolucijsko programiranje ne uporablja križanja. Geni kromosomov so tipično realna števila. Vsakemu kromosomu je dodana množica parametrov mutacije. Teh parametrov je toliko, kolikor je spremenljivk, ki jih optimiramo. Podobno kot evolucijske strategije tudi evolucijsko programiranje uporablja Gaussovo porazdelitev s povprečno vrednostjo 0 in standardno deviacijo, ki je parameter mutacije, za spreminjanje oziroma mutiranje genov. Genu prištejemo naključno izbrano vrednost na Gaussovi krivulji. Za spremembo od evolucijskih strategij pa tu uporabljamo za vsako spremenljivko svojo standardno deviacijo. Te deviacije tudi optimiramo, saj so shranjene v kromosomih skupaj z drugimi spremenljivkami, čemur pravimo samoprilagajanje [2].

Izbira naslednje populacije poteka tako, da vsakega izmed staršev in potomcev primerjamo z določenim številom naključno izbranih osebkov ter preštejemo, v koliko primerih je izbrani osebek boljši. Naslednja populacija je množica osebkov z največ boljšimi ocenitvami.

3.2.4 Genetsko programiranje

Genetsko programiranje (GP) uporablja drevesno predstavitev kromosomov. Za razliko od ostalih evolucijskih algoritmov, ki so optimizacijski algoritmi, lahko GP uvrstimo med algoritme strojnega učenja. Ostali evolucijski algoritmi iščejo take rešitve, ki dajejo najvišjo oceno pri znanem načinu ocenjevanja. Genetsko programiranje pa gradi model, ki najbolj ustreza danim podatkom. Ustrezanje podatkom ocenjujemo s predpisanim postopkom ter tako dobimo oceno oziroma stopnjo ustreznosti, ki jo želimo maksimizirati. Tako lahko tudi genetsko programiranje uvrstimo med optimizacijske algoritme [3].

Kromosomi so nelinearna struktura, katere velikost ni fiksna. To je predvsem posledica različnih velikosti dreves, ki predstavljajo kromosome. Predstavimo jih lahko kot aritmetične ali logične formule. Ker lahko programsko kodo predstavimo kot drevo oziroma formulo, lahko tudi programe optimiramo z genetskim programiranjem. Postopek se imenuje avtomatska evolucija računalniških programov.

Pri križanju naključno izberemo dve vozlišči v drevesu, eno za vsakega starša. Nato izmenjamo poddrevesi, katerih korena sta izbrani vozlišči. Parameter križanja je verjetnost izbora vozlišča za križanje.

Mutacijo drevesa izvedemo tako, da izberemo naključno vozlišče v drevesu ter poddrevo tega vozlišča zamenjamo z naključno tvorjenim drevesom. Verjetnost mutacije je tipično nizka.

3.2.5 Diferencialna evolucija

Pri diferencialni evoluciji so kromosomi predstavljeni kot vektorji realnih števil. Velikost populacije je M_{de}. Vsak kromosom ima enolično določeno število, imenovano indeks kromosoma. Indeksi so zaporedna števila. Ker so kromosomi vektorji, lahko nad njimi izvajamo vektorske operacije.

Mutiramo vse kromosome populacije. Mutacijo naključno izbranega kromosoma z zaporedno številko m_{de} izvedemo tako, da naključno izberemo dva druga kromosoma ter določimo vektorsko razliko med njima. To pomnožimo z naključno izbranim pozitivnim številom in prištejemo kromosomu, ki je bil izbran za mutiranje, ter dobimo mutiran kromosom. Če lahko pri naključni

izbiri dveh kromosomov izberemo dvakrat isti kromosom, potem se mutacija ne izvede. To se zgodi z verjetnostjo $\frac{1}{M_{de}}$. Lahko pa določimo, da ni možna dvakratna izbira istega kromosoma. V tem primeru se mutacija vedno izvede.

Po mutaciji izvedemo križanje med kromosomom iz populacije z indeksom m_{de} in mutiranim kromosomom. Za vsak gen novega kromosoma določimo, ali bo imel vrednost gena mutiranega kromosoma ali vrednost gena kromosoma z indeksom m_{de}. Verjetnost dodelitve vrednosti gena mutiranega kromosoma je parameter algoritma. Poleg tega naključno izbranemu genu vedno dodelimo vrednost gena mutiranega kromosoma ter tako zagotovimo, da nov kromosom ni nikoli enak kromosomu z indeksom m_{de}.

Po mutaciji in križanju ocenimo nov kromosom in kromosom z indeksom m_{de}. Kromosom z boljšo oceno ostane v populaciji. Njegov indeks je m_{de} [13].

3.3 Uglaševanje evolucijskih algoritmov

Prilagajanje algoritma dani nalogi je določanje spremenljivk in parametrov evolucijskega algoritma tako, da čim bolj ustrezajo dani nalogi [6]. Z določanjem parametrov vnesemo določeno znanje v algoritem, poleg tega pa lahko omogočimo prilagajanje parametrov glede na problem [12]. Prilagajanje delimo glede na [8]:

- tip prilagajanja:
  - statično,
  - dinamično:
    - deterministično,
    - prilagodljivo,
    - samoprilagodljivo,
- nivo prilagajanja:
  - okolje,
  - populacija,
  - posameznik,
  - komponenta.

Pri statičnem prilagajanju imajo parametri stalno vrednost skozi ves evolucijski proces iskanja rešitev. Vrednosti parametrov določimo po večkratnem

poganjanju algoritma [7]. Primer iskanja parametrov je metaevolucijski algoritem. To je evolucijski algoritem, ki ima v kromosomu parametre osnovnega evolucijskega algoritma. Ocenjevanje kromosomov izvedemo s poganjanjem osnovnega evolucijskega algoritma pri parametrih, določenih v kromosomu.

Dinamično prilagajanje izvajamo brez zunanjega nadzora.

Pri deterministčnem dinamičnem prilagajanju parametre spreminjamo v skladu z deterministčnim pravilom, brez upoštevanja povratne informacije, ponavadi po izvedbi določenega števila generacij od zadnjega prilagajanja.

Pri prilagodljivem dinamičnem prilagajanju uporabimo povratno informacijo za spremembo parametrov.

Pri samoprilagodljivem dinamičnem prilagajanju ali evoluciji evolucije so parametri shranjeni v kromosomih, kar pomeni, da jih mutiramo in križamo skupaj z ostalimi vrednostmi v kromosomih. Vsak kromosom ima tako lastne parametre algoritma, na primer lastno verjetnost križanja, lastno verjetnost mutacije itd. [11].

Pri prilagajanju na nivoju okolja spreminjamo odziv okolja na posameznike, na primer sprememba uteži v ocenitveni funkciji.

Pri prilagajanju na nivoju populacije gre za spreminjanje globalnih parametrov, ki vplivajo na vse člane populacije, na primer sprememba verjetnosti mutacije.

Pri prilagajanju na nivoju posameznika spreminjamo parametre, ki vplivajo na vsakega posameznika posebej, na primer prilagajanje mest križanja glede na posameznika.

S prilagajanjem na nivoju komponent spreminjamo parametre, ki vplivajo na posamezen gen, na primer verjetnost mutacije posameznega gena.

3.4 Področja uporabe

Evolucijske algoritme uporabljamo na področjih, kjer so relacije med spremenljivkami neznane ali pa jih le delno poznamo, kjer matematika ne nudi analitičnih rešitev ali pa obstajajo analitične rešitve, katerih čas izvedbe ni sprejemljiv, kjer je sprejemljiva tudi aproksimacijska rešitev, kjer je potrebno obdelati in klasificirati veliko množico podatkov itd. [3].

Primeri problemov, za reševanje katerih lahko uporabimo evolucijske algoritme, so:

- problem trgovskega potnika,
- navigacija v prostoru,

- sestavljanje urnika,
- usmerjanje umetne mravlje,
- simbolična regresija,
- učenje kontekstno prostih gramatik [10],
- strojno učenje,
- uglaševanje parametrov,
- avtomatsko programiranje računalnikov [8],
- rudarjenje podatkov,
- klasifikacija slik [3].

Poglavje 4

Evolucijski algoritem za optimiranje prevoza tovora

4.1 Opis algoritma

Problem optimalnega prevoza tovora smo razdelili na več podproblemov. Vsak podproblem obsega en cikel. Prevoz tovora v enem ciklu optimiramo z evolucijskim algoritmom tako, da poskušamo najti čim boljši razpored tovora na vozila in čim boljši razpored vozil na nakladalna in razkladalna mesta. Po vsakem ciklu zmanjšamo ves tovor za tovor, prepeljan v tem ciklu. Cikle izvajamo, dokler ostaja še kaj tovora za prevoz. Čas prevoza vsega tovora je enak vsoti časov, porabljenih za prevoz tovora po ciklih.

Za optimiranje vsakega cikla posebej smo se odločili, ker bi bila predstavitev rešitev v obliki kromosomov za celoten problem prezahtevna. Ker v naprej ne poznamo števila ciklov, ki jih bo potrebno izvesti za prevoz vsega tovora, ne moremo določiti ustrezne dolžine kromosoma. V populaciji bi tako bili kromosomi različnih dolžin, kar je neprikladno za uporabo genetskih operatorjev.

4.2 Postopek reševanja

Iščemo najboljšo rešitev za vsak cikel posebej. Poiščemo jo z evolucijskim algoritmom. Na začetku tvorimo začetno populacijo V_{pop} rešitev. Te rešitve tvorimo naključno tako, da vsakemu vozilu naključno dodelimo vrsto tovora, nakladalno mesto in razkladalno mesto. Naključno tvorjene rešitve so lahko nedopustne. Če je rešitev nedopustna, jo popravimo. Rešitev je nedopustna, če smo kakšnemu vozilu dodelili vrsto tovora, ki ga ne more voziti ali pa je bil

že v celoti prepeljan. Poleg tega je lahko vozilu dodeljeno nakladalno oziroma razkladalno mesto, na katerem ni mogoče nakladati oziroma razkladati vrste tovora, ki je dodeljena vozilu. Popravljanje rešitev je opisano v razdelku 4.9.

Nato iščemo boljše rešitve v iterativnih korakih (generacijah). V vsaki generaciji ($V_{pop}/2$)-krat izvedemo:

1. turnirsko izbiro dveh rešitev (glej razdelek 4.5),
2. križanje izbranih dveh rešitev (razdelek 4.6).
3. naključno spremembo (mutacijo) rešitev, dobljenih s križanjem (razdelek 4.7),
4. popravljanje tako dobljenih dveh novih rešitev zaradi morebitne nedopustnosti (razdelek 4.9),
5. vrednotenje novih rešitev z ocenitveno funkcijo,
6. vstavljanje dveh najboljših rešitev izmed štirih (dve prvotno izbrani in dve iz njiju dobljeni) v populacijo.

4.3 Predstavitev rešitev

Posamezna rešitev ali kromosom definira značilnosti vozil v posameznem ciklu. Zapisana je v obliki vektorja n genov, kjer vsak gen predstavlja eno izmed vozil. Gen, ki predstavlja i-to vozilo v ciklu j, vsebuje naslednje podatke:

- u_{ij} – vrsta tovora,
- x_{ij} – indeks nakladalnega mesta,
- y_{ij} – indeks razkladalnega mesta.

Vrsta tovora je zapisana s celim številom. Možne vrednosti so $1, \ldots, v$, ker imamo v različnih vrst tovora. Dovoljene vrednosti pa so le tiste, ki predstavljajo vrste tovora, ki jih lahko vozi določeno vozilo, oziroma za vozilo i so dovoljene tiste vrste tovora j, kjer je $w_{ij} > 0$. Poleg teh vrednosti je za vrsto tovora dovoljena tudi vrednost 0, ki pomeni, da vozilo ne vozi nobene vrste tovora, ker so bile vse vrste tovora, ki jih lahko vozi vozilo, že predhodno v celoti prepeljane. V takem primeru vozilo ne vozi v trenutnem ciklu oziroma vozilo ostane v kraju A.

Indeks nakladalnega mesta je celo število. Možne vrednosti so $1, \ldots, m$, ker imamo m nakladalnih mest. Dovoljene vrednosti pa so le tiste, ki predstavljajo nakladalna mesta, na katerih se lahko naklada tovor, dodeljen vozilu, oziroma če je vozilu dodeljen tovor i, potem so dovoljena tista nakladalna mesta j, kjer je $a_{ji} > 0$. Poleg teh vrednosti je za nakladalno mesto možna tudi vrednost 0, ki pomeni, da se vozilo samo naklada. Ta vrednost pa je dovoljena le, če ima vozilo samonakladalno napravo in ta naprava lahko naklada dodeljeno vrsto tovora, oziroma za vozilo i z dodeljeno vrsto tovora j je dovoljeno samonakladanje, če je $a^S_{ij} > 0$.

Indeks razkladalnega mesta je celo število. Možne vrednosti so $1, \ldots, k$, ker imamo k razkladalnih mest. Dovoljene vrednosti pa so le tiste, ki predstavljajo razkladalna mesta, na katerih se lahko razklada vozilu dodeljen tovor, oziroma če je vozilu dodeljen tovor i, potem so dovoljena tista razkladalna mesta j, kjer je $b_{ji} > 0$. Poleg teh vrednosti je za razkladalno mesto možna tudi vrednost 0, ki pomeni, da se vozilo samo razklada. Ta vrednost pa je dovoljena le, če ima vozilo samonakladalno napravo in ta naprava lahko razklada dodeljeno vrsto tovora, oziroma za vozilo i z dodeljeno vrsto tovora j je dovoljeno samorazkladanje, če je $b^S_{ij} > 0$.

4.4 Vrednotenje rešitev

Cilj algoritma je najti tako prireditev vrst tovora ter nakladalnih in razkladalnih mest vozilom, da bo skupni čas prevoza vsega tovora najkrajši.

Glede na formulacijo algoritma pa je potrebno redefinirati oceno rešitev, saj z evolucijskim algoritmom rešujemo le del prevoza, le en cikel. Pri tem pa ne more biti cilj čim krajši čas izvedbe tega cikla. To bi pomenilo, da bi na začetku nakladali tiste vrste tovora, ki porabijo najmanj časa za nakladanje in razkladanje. Pri kasnejših ciklih pa bi nakladali ostale tovore, ki zahtevajo več časa, s čimer bi se število ciklov, potrebnih za prevoz vsega tovora, povečalo.

Kot primer vzemimo, da imamo deset vozil in dve vrsti tovora. Vseh deset vozil lahko vozi prvo vrsto tovora, le eno pa drugo vrsto tovora, poleg tega pa je nosilnost tega vozila za drugo vrsto tovora zelo majhna in tudi hitrosti nakladanja in razkladanja tega tovora sta majhni. V takem primeru ni smiselno na začetku na vseh deset vozil naložiti prve vrste tovora, ko pa tega zmanjka, pa bo eno samo vozilo prevažalo drugo vrsto tovora. Ker želimo algoritem, ki bo splošen, je ocena samo na podlagi časa neustrezna.

Rešitve vrednotimo z ocenitveno funkcijo O, ki poleg časa cikla upošteva tudi prevožen tovor. Količina tovora, naloženega na vozilo l, je U_l. Skupni čas

nakladanja vozil v koloni je

$$T_N = \max \left\{ \max_{i=1,\ldots,m} \sum_{\substack{l=1 \\ x_{lj}=i}}^{n} \frac{U_l}{a_{i,u_{lj}}}, \max_{\substack{l=1,\ldots,n \\ x_{lj}=0}} \frac{U_l}{a^s_{l,u_{lj}}} \right\}, \tag{4.1}$$

skupen čas razkladanja pa

$$T_R = \max \left\{ \max_{i=1,\ldots,k} \sum_{\substack{l=1 \\ y_{lj}=i}}^{n} \frac{U_l}{b_{i,u_{lj}}}, \max_{\substack{l=1,\ldots,n \\ y_{lj}=0}} \frac{U_l}{b^s_{l,u_{lj}}} \right\}. \tag{4.2}$$

Čas za izvedbo enega cikla je potem

$$T = T_N + \max_{\substack{i=1,\ldots,n \\ U_i>0}} t_i^{(1)} + T_R + \max_{\substack{i=1,\ldots,n \\ U_i>0}} t_i^{(0)}. \tag{4.3}$$

Poleg minimizacije časa cikla želimo maksimizacijo prevoženega tovora. Pri tem pa ne moremo vzeti vsote količin vseh vrst tovora na vozilih, ki smo jih prepeljali v ciklu, saj vsako vrsto tovora merimo v lastni merski enoti. Zato vzamemo vsoto relativnih vrednosti količin vseh vrst tovora na vozilih glede na celotne količine ustreznih vrst tovora. To vsoto želimo maksimizirati, zato se nahaja v števcu ocene. Čas želimo minimizirati, zato se nahaja v imenovalcu ocene.

Ocena rešitve je

$$O = \frac{1}{T} \sum_{l=1}^{n} \frac{U_l}{V_{u_{lj}}}. \tag{4.4}$$

4.5 Selekcija

V algoritmu je uporabljena turnirska selekcija [7]. Velikost turnirja V_{tur} je parameter algoritma.

S turnirsko selekcijo izberemo dva kromosoma. Na začetku naključno izberemo V_{tur} kromosomov. Izmed teh kromosomov izberemo najboljšega glede na ocene kromosomov. Ta kromosom je prvi starš. Potem znova izberemo V_{tur} kromosomov, med katerimi ne sme biti prvi starš. Izmed teh kromosomov izberemo najboljšega glede na ocene kromosomov. Ta kromosom je drugi starš. Oba starša podvojimo ter tako dobimo dva potomca. Nad potomcema izvedemo križanje in mutacijo.

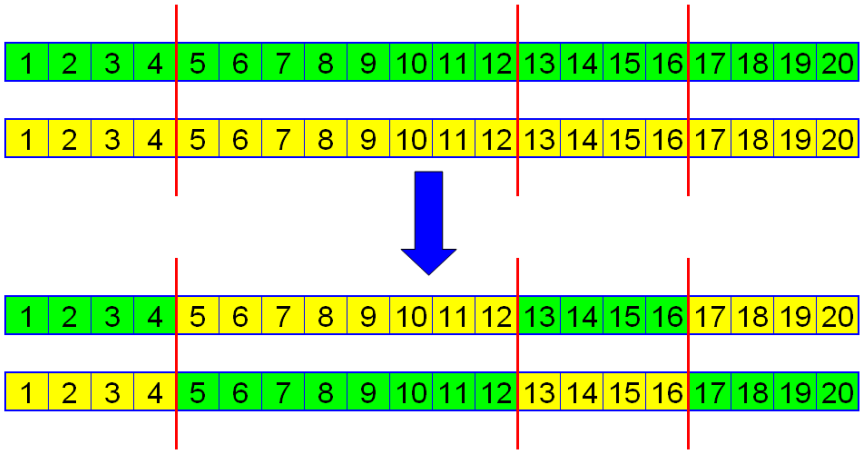
Po izvedenem križanju, mutaciji in popravljanju izberemo najboljša dva izmed staršev in potomcev glede na njihovo oceno. Če je kateri izmed teh dveh najboljših kromosomov potomec, potem ga dodamo populaciji. Če kateri izmed staršev ni med dvema najboljšima kromosomoma, potem ga zbrišemo iz populacije.

4.6 Križanje

V algoritmu je uporabljeno večmestno križanje. Število mest križanja M_k in verjetnost križanja p_k sta parametra algoritma.

Izbrana dva kromosoma križamo z verjetnostjo p_k. Ko se izvede križanje, naključno izberemo M_k mest križanja. Mesto križanja mora biti med 1 in $n-1$, kjer je n število genov. Križanje izvedemo tako, da istoležne komponente kromosomov med vsakim drugim zaporednim parom mest križanja zamenjamo med seboj.

Kot primer si lahko ogledamo trimestno križanje, ko imamo 20 vozil, označenih z indeksi od 1 do 20. Izbrana mesta križanja so 4, 12 in 16. V tem primeru zamenjamo komponente dveh kromosomov z indeksi od 5 do 12 ter od 17 do 20. Komponent z indeksi od 1 do 4 ter od 13 do 16 ne zamenjamo. Križanje je prikazano na sliki 4.1.



Slika 4.1: Primer križanja.

Za vsak par kromosomov posebej naključno izberemo mesta križanja. Pri zamenjavi komponent zamenjamo vse podatke, ki se nahajajo v komponentah oziroma genih. Ti podatki so vrsta tovora, indeks nakladalnega mesta in indeks razkladalnega mesta.

4.7 Mutacija

Verjetnost mutiranja p_m je parameter algoritma. Vsak gen vsakega izmed dveh izbranih kromosomov mutiramo z verjetnostjo p_m. Ko določen gen mutiramo, mu določimo naključne vrednosti za vse podatke, ki jih vsebuje. Genu določimo naključno vrsto tovora med 1 in v, naključen indeks nakladalnega mesta med 1 in m ter naključen indeks razkladalnega mesta med 1 in k.

4.8 Ustavitveni pogoj

Postopek, ki smo ga opisali v razdelku 4.2, izvajamo največ N generacij oziroma ga ustavimo, če zadnjih K generacij ni bilo izboljšanja najboljše rešitve. Na koncu vrnemo najboljšo rešitev.

4.9 Popravljanje rešitev

Nedopustne rešitve lahko nastanejo ob začetnem naključnem tvorjenju kromosomov ter ob spremembi obstoječih kromosomov s križanjem oziroma z mutacijo. Pri tem ima lahko vsak izmed treh podatkov o vozilu nedopustno vrednost.

Vozilo i ima dodeljeno nedopustno vrsto tovora, če ne more voziti te vrste tovora, kar pomeni, da je $w_{ij} = 0$, oziroma če je bila količina dodeljene vrste tovora že v celoti prepeljana.

Vozilo i, kateremu je dodeljena vrsta tovora j, ima dodeljeno nedopustno nakladalno mesto k, če na njem ne moremo nakladati vrste tovora j, kar pomeni, da je $a_{kj} = 0$.

Vozilo i, kateremu je dodeljena vrsta tovora j, ima dodeljeno nedopustno razkladalno mesto k, če na njem ne moremo razkladati vrste tovora j, kar pomeni, da je $b_{kj} = 0$.

Vsako nedopustno vrednost, dodeljeno vozilu, je potrebno popraviti. Popravljanje komponent rešitev izvajamo v naključnem vrstnem redu. Najprej preverimo, če je vozilu dodeljena vrsta tovora, ki ga lahko vozi ter je še na voljo. Če to ne velja, potem vozilu dodelimo naključno vrsto tovora, ki ga lahko vozi ter ga je še na voljo in ga niso v celoti prepeljala že predhodno obravnavana vozila.

Nato po potrebi popravimo še indeksa nakladalnega in razkladalnega mesta. Če ima vozilo samonakladalno napravo, ki lahko naklada dodeljeno vrsto tovora, potem mu dodelimo samonakladanje z verjetnostjo $p_s = \frac{1}{1+S_N}$, kjer je

S_N število nakladalnih mest, na katerih je mogoče nakladati dodeljeno vrsto tovora i, oziroma $S_N = |a_{ij}|_{a_{ij}>0}$. Če ima vozilo samonakladalno napravo, ki lahko razklada dodeljeno vrsto tovora, potem mu dodelimo samorazkladanje z verjetnostjo $p_s = \frac{1}{1+S_R}$, kjer je S_R število razkladalnih mest, na katerih je mogoče razkladati dodeljeno vrsto tovora i, oziroma $S_R = |b_{ij}|_{b_{ij}>0}$. Če vozilu ni dodeljeno samonakladanje in na dodeljenem nakladalnem mestu ne moremo nakladati dodeljene vrste tovora, potem vozilu naključno dodelimo nakladalno mesto, na katerem lahko nakladamo dodeljeno vrsto tovora. Če vozilu ni dodeljeno samorazkladanje in na dodeljenem razkladalnem mestu ne moremo razkladati dodeljene vrste tovora, potem vozilu naključno dodelimo razkladalno mesto, na katerem lahko razkladamo dodeljeno vrsto tovora.

Poglavje 5

Poskusi in rezultati

5.1 Opis testnih problemov

Razviti evolucijski algoritem smo testirali na štirih testnih problemih prevoza tovora. Te smo definirali tako, da imajo različne lastnosti ter zato različne zahtevnosti reševanja. Pri prvem testnem problemu je treba prepeljati 3 vrste tovora, katerih količine so prikazane v tabeli 5.1.

V_1	V_2	V_3
30	1000	200000

Tabela 5.1: Količine posameznih vrst tovora.

V kraju A je 5 nakladalnih mest, za katera so hitrosti nakladanja prikazane v tabeli 5.2.

a_{i1}	a_{i2}	a_{i3}
4	30	5000
0	50	5000
0	0	40000
4	35	4000
0	40	7000

Tabela 5.2: Hitrosti nakladanja vrst tovora po posameznih nakladalnih mestih.

V kraju B so 4 razkladalna mesta, za katera so hitrosti razkladanja prikazane v tabeli 5.3.

Na voljo imamo 20 vozil, katerih lastnosti so prikazane v tabeli 5.4.

a_{i1}	a_{i2}	a_{i3}
4	30	10000
0	50	5000
0	0	50000
3	35	4000

Tabela 5.3: Hitrosti razkladanja vrst tovora po posameznih razkladalnih mestih.

Nosilnost			Hitrost samonakladanja			Hitrost samorazkladanja			Čas vožnje		Število vozil
w_{i1}	w_{i2}	w_{i3}	a_{i1}^S	a_{i2}^S	a_{i3}^S	b_{i1}^S	b_{i2}^S	b_{i3}^S	$t_i^{(1)}$	$t_i^{(0)}$	
2	38	15000	0	0	10	0	0	10	1,04	1,01	2
2	50	20000	0	0	10	0	0	10	1,13	1,04	4
0	12	5000	0	20	10	0	20	10	1,01	1,01	6
1	8	10	2	0	10	2	0	10	1,01	1,01	4
0	8	3000	0	0	10	0	0	10	1,01	0,98	4

Tabela 5.4: Lastnosti vozil.

Pri drugem testnem problemu je treba prepeljati tovor, prikazan v tabeli 5.1. V tabeli 5.2 so prikazana nakladalna mesta. Razkladalna mesta smo prikazali v tabeli 5.3. V tabeli 5.5 smo prikazali lastnosti vozil.

Nosilnost			Hitrost samonakladanja			Hitrost samorazkladanja			Čas vožnje		Število vozil
w_{i1}	w_{i2}	w_{i3}	a_{i1}^S	a_{i2}^S	a_{i3}^S	b_{i1}^S	b_{i2}^S	b_{i3}^S	$t_i^{(1)}$	$t_i^{(0)}$	
2	38	0	0	0	10	0	0	10	1,04	1,01	1
2	50	20000	0	0	10	0	0	10	1,13	1,04	1
0	8	10	2	0	10	2	0	10	1,01	1,01	2

Tabela 5.5: Lastnosti vozil v drugem testnem problemu.

Pri tretjem testnem problemu je treba prepeljati tovor, prikazan v tabeli 5.1. V tabeli 5.6 so prikazana nakladalna mesta. Razkladalna mesta smo prikazali v tabeli 5.7. V tabeli 5.4 smo prikazali lastnosti vozil.

Pri četrtem testnem problemu je treba prepeljati tovor, prikazan v tabeli 5.8. V tabeli 5.2 so prikazana nakladalna mesta. Razkladalna mesta smo prikazali v tabeli 5.3. V tabeli 5.4 smo prikazali lastnosti vozil.

a_{i1}	a_{i2}	a_{i3}
4	30	5000
0	50	50
0	0	40000
40	3	4000
0	40	7000
0	40	0
67	0	0
45	40	7000
0	0	7000

Tabela 5.6: Hitrosti nakladanja vrst tovora po posameznih nakladalnih mestih v tretjem testnem problemu.

a_{i1}	a_{i2}	a_{i3}
10	30	10
0	50	5000

Tabela 5.7: Hitrosti razkladanja vrst tovora po posameznih razkladalnih mestih v tretjem testnem problemu.

V_1	V_2	V_3
35	1800	200

Tabela 5.8: Količine posameznih vrst tovora v četrtem testnem problemu.

5.2 Poskusi in rezultati z začetnimi nastavitvami parametrov algoritma

Parametrom algoritma smo po testiranju na testnih problemih optimizacijske naloge določili naslednje privzete vrednosti:

- velikost populacije $V_{pop} = 100$,
- maksimalno število generacij $N = 200$,
- število generacij, po katerih algoritem ustavimo, če ne pride do izboljšanja rešitve, $K = 40$,
- velikost turnirja $V_{tur} = 14$,
- verjetnost križanja $p_k = 0{,}9$,

- število mest križanja $M_k = 2$,
- verjetnost mutacije $p_m = 0{,}1$.

Za vsak problem smo 10-krat pognali algoritem. Rezultati so statistično obdelani in prikazani v tabeli 5.9. Opazimo lahko, da je algoritem vrnil dobre rezultate za prvi, drugi in četrti testni problem, saj je standardni odklon majhen. Pri tretjem testnem problemu pa je standardni odklon velik, saj je ta problem zahtevnejši od ostalih in zato je manjša ponovljivost rezultatov. Algoritem je rezultate vrnil v manj kot 10 sekundah, kar je sprejemljiv čas za uporabnika.

Testni problem	Minimalni čas	Povprečni čas	Maksimalni čas	Standardni odklon
1	30,42	31,76	33,17	0,93
2	74,38	74,39	74,42	0,10
3	348,58	526,62	641,42	86,13
4	45,30	46,04	46,92	0,41

Tabela 5.9: Rezultati testnih problemov.

5.3 Metaevolucijski algoritem

5.3.1 Splošno o metaevolucijskih algoritmih

Metaevolucijski algoritem je evolucijski algoritem, ki optimira delovanje evolucijskega algoritma. Z evolucijskim algoritmom iščemo vrednosti spremenljivk, katerih ocena je najboljša glede na ocenitveno funkcijo. Način iskanja in kakovost najdenih rešitev lahko spreminjamo z vrednostmi parametrov evolucijskega algoritma. Potrebno je najti optimalne vrednosti teh parametrov.

Vrednosti parametrov lahko uglasimo na več načinov. Eden izmed njih je uporaba metaevolucijskega algoritma. Delovanje tega algoritma je enako delovanju evolucijskega algoritma. Parametre evolucijskega algoritma shranimo v kromosom metaevolucijskega algoritma. Oceno kromosoma dobimo z izvedbo evolucijskega algoritma z vrednostmi parametrov, ki so določene s kromosomom. Iščemo optimum te ocene.

Z izvajanjem metaevolucijskega algoritma določimo vrednosti parametrov, s katerimi kasneje izvajamo evolucijski algoritem.

5.3.2 Predstavitev rešitev v metaevolucijskem algoritmu

Velikost populacije je 200 kromosomov. Vsak kromosom vsebuje parametre evolucijskega algoritma. Zapisan je z vektorjem dolžine 9. Vsak gen predstavlja en parameter. Parametri oziroma geni kromosoma so:

- ocenitvena funkcija,
- potenca ocenitvene funkcije,
- velikost populacije,
- velikost turnirja,
- maksimalno število generacij,
- število generacij brez izboljšav,
- verjetnost križanja,
- število mest križanja,
- verjetnost mutacije.

Med testiranjem evolucijskega algoritma smo definirali štiri ocenitvene funkcije. Splošna oblika funkcij je:

$$O = \frac{P_{tovor}}{T}, \tag{5.1}$$

kjer je P_{tovor} odstotek prevoženega tovora in je različen glede na ocenitveno funkcijo, T pa je čas cikla in je pri vseh ocenitvenih funkcijah enak.

Skupni čas nakladanja vozil je

$$T_N = \max\left\{ \max_{i=1,\ldots,m} \sum_{\substack{l=1 \\ x_{lj}=i}}^{n} \frac{U_l}{a_{i,u_{lj}}}, \max_{\substack{l=1,\ldots,n \\ x_{lj}=0}} \frac{U_l}{a^s_{l,u_{lj}}} \right\}, \tag{5.2}$$

skupni čas razkladanja pa

$$T_R = \max\left\{ \max_{i=1,\ldots,k} \sum_{\substack{l=1 \\ y_{lj}=i}}^{n} \frac{U_l}{b_{i,u_{lj}}}, \max_{\substack{l=1,\ldots,n \\ y_{lj}=0}} \frac{U_l}{b^s_{l,u_{lj}}} \right\}. \tag{5.3}$$

Čas za izvedbo enega cikla je potem

$$T = T_N + \max_{\substack{i=1,\dots,n \\ U_i>0}} t_i^{(1)} + T_R + \max_{\substack{i=1,\dots,n \\ U_i>0}} t_i^{(0)}. \tag{5.4}$$

Količina vrste tovora i, ki jo je potrebno še prepeljati iz kraja A v kraj B v trenutnem ciklu, je $V_{i,ost}$. Pri vsaki ocenitveni funkciji uporabimo potenco ocenitvene funkcije Z, s katero potenciramo odstotke količin prevožene vrste tovora. S povečevanjem Z damo večjo težo večjim odstotkom, kar zmanjšuje njihovo enakomernost.

Delež prevožene vrste tovora, ki ga uporabimo v prvi funkciji, je:

$$P_{tovor} = \sum_{l=1}^{n} (\frac{U_l}{V_{u_{lj},ost}})^Z. \tag{5.5}$$

Delež prevožene vrste tovora, ki ga uporabimo v drugi funkciji, je:

$$P_{tovor} = \sum_{l=1}^{n} (\frac{U_l}{V_{u_{lj}}})^Z. \tag{5.6}$$

V tretji in četrti ocenitveni funkciji upoštevamo le tisto vrsto tovora q, katere je potrebno še največ prepeljati:

$$q = \max_{i=1}^{v} \frac{V_{i,ost}}{V_i}. \tag{5.7}$$

Delež prevožene vrste tovora, ki ga uporabimo v tretji funkciji, je:

$$P_{tovor} = \sum_{\substack{l=1 \\ u_{lj}=q}}^{n} (\frac{U_l}{V_{u_{lj},ost}})^Z. \tag{5.8}$$

Delež prevožene vrste tovora, ki ga uporabimo v četrti funkciji, je:

$$P_{tovor} = \sum_{\substack{l=1 \\ u_{lj}=q}}^{n} (\frac{U_l}{V_{u_{lj}}})^Z. \tag{5.9}$$

Ocenitvena funkcija je shranjena kot celo število, katerega vrednost je med 1 in 4, ker imamo 4 različne ocenitvene funkcije. Potenca ocenitvene funkcije je shranjena kot celo število, katerega vrednost je med 1 in 20. Velikost populacije je shranjena kot celo število, katerega vrednost je med 50 in 200 s korakom 10. Velikost turnirja je shranjena kot celo število, katerega vrednost je med

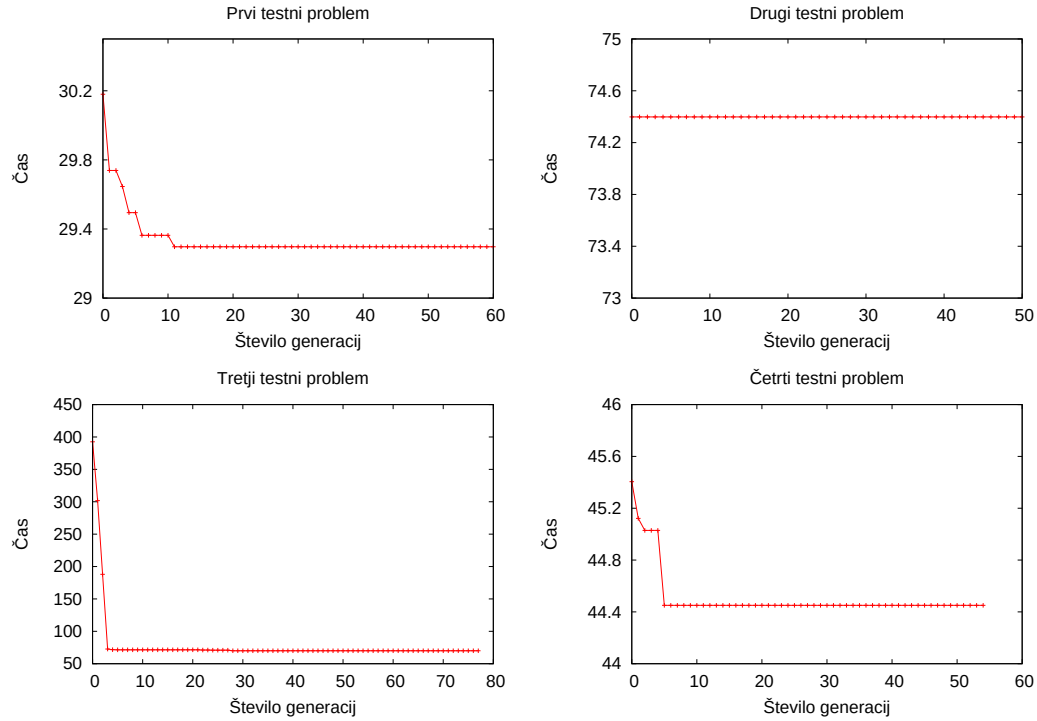
2 in velikostjo populacije. Maksimalno število generacij je shranjeno kot celo število, katerega vrednost je med 50 in 200 s korakom 10. Število generacij brez izboljšav je shranjeno kot celo število, katerega vrednost je enaka petini maksimalnega števila generacij. Verjetnost križanja je shranjena kot realno število, katerega vrednost je med 0,5 in 1 s korakom 0,1. Število mest križanja je shranjeno kot celo število, katerega vrednost je med 1 in $n - 1$, kjer je n število vozil. Verjetnost mutacije je shranjena kot realno število, katerega vrednost je med 0 in 0,2 s korakom 0,05.

5.3.3 Ostale značilnosti metaevolucijskega algoritma

Rešitev oziroma kromosom ocenimo tako, da evolucijski algoritem petkrat poženemo z vrednostmi parametrov, ki so shranjene v kromosomu, in kot oceno vzamemo povprečen čas voženj. Iščemo kromosom z najnižjo oceno. Uporabili smo turnirsko selekcijo, ki je opisana v razdelku 4.5. Velikost turnirja je 20. Kromosoma križamo z verjetnostjo 0,9. Število mest križanja je 3. Gen kromosoma mutiramo z verjetnostjo 0,1. Določimo mu naključno vrednost iz njegove zaloge vrednosti. Zaloge vrednosti genov so opisane v razdelku 5.3.2. Izvajanje algoritma ustavimo, če se zadnjih 50 generacij ocena najboljšega kromosoma ne spremeni. Kromosome popravljamo, saj lahko križani oziroma mutirani kromosomi vsebujejo nedopustne vrednosti. Velikost turnirja ne sme biti večja od velikosti populacije. Če je večja, potem se za velikost turnirja izbere naključno vrednost med 2 in velikostjo populacije. Število generacij brez izboljšav nastavimo vedno na petino maksimalnega števila generacij.

5.3.4 Rezultati poskusov z metaevolucijskim algoritmom

Algoritmu smo z metaevolucijskim algoritmom določili vrednosti parametrov. Spreminjanje časa prevoza vsega tovora je prikazano na sliki 5.1. Čas prevoza prvega tovora se je med izvajanjem algoritma v 61 generacijah izboljšal za 3% in je 29,30 ur. Čas prevoza drugega tovora se med izvajanjem algoritma v 51 generacijah ni izboljšal in je 74,40 ur. Čas prevoza tretjega tovora se je med izvajanjem algoritma v 78 generacijah izboljšal za 461% in je 70,01 ur. Čas prevoza četrtega tovora pa se je med izvajanjem algoritma v 55 generacijah izboljšal za 2% in je 44,45 ur. Vrednosti parametrov algoritma so prikazane v tabeli 5.10.



Slika 5.1: Spreminjanje časa prevoza tovora skozi generacije metaevolucijskega algoritma.

Parametri evolucijskega algoritma	Številka problema			
	1	2	3	4
Ocenitvena funkcija	2	2	2	2
Potenca ocenitvene funkcije	1	1	1	1
Velikost populacije	190	90	140	150
Velikost turnirja	10	10	18	18
Maksimalno število generacij	200	110	200	130
Število generacij brez izboljšav	40	22	40	26
Verjetnost križanja	0,6	0,8	0,9	0,5
Število mest križanja	13	1	19	2
Verjetnost mutacije	0,03	0,13	0,0	0,03

Tabela 5.10: Vrednosti parametrov algoritma, dobljene z reševanjem testnih problemov.

5.4 Primerjava rezultatov testiranja z rezultati drugih metod

Na Fakulteti za matematiko in fiziko Univerze v Ljubljani so razvili požrešni algoritem, ki rešuje problem optimiranja prevoza tovora med dvema lokacijama

s skupino vozil [5]. V tabeli 5.11 so prikazani rezultati, dobljeni z evolucijskim algoritmom, in rezultati, dobljeni s požrešno metodo.

Številka problema	Povprečni rezultat evolucijskega algoritma v urah	Povprečni rezultat uglašenega algoritma v urah	Rezultat požrešne metode v urah
1	31,76	29,08	32,25
2	74,39	74,39	79,99
3	526,62	70,01	68,63
4	46,04	44,45	53,79

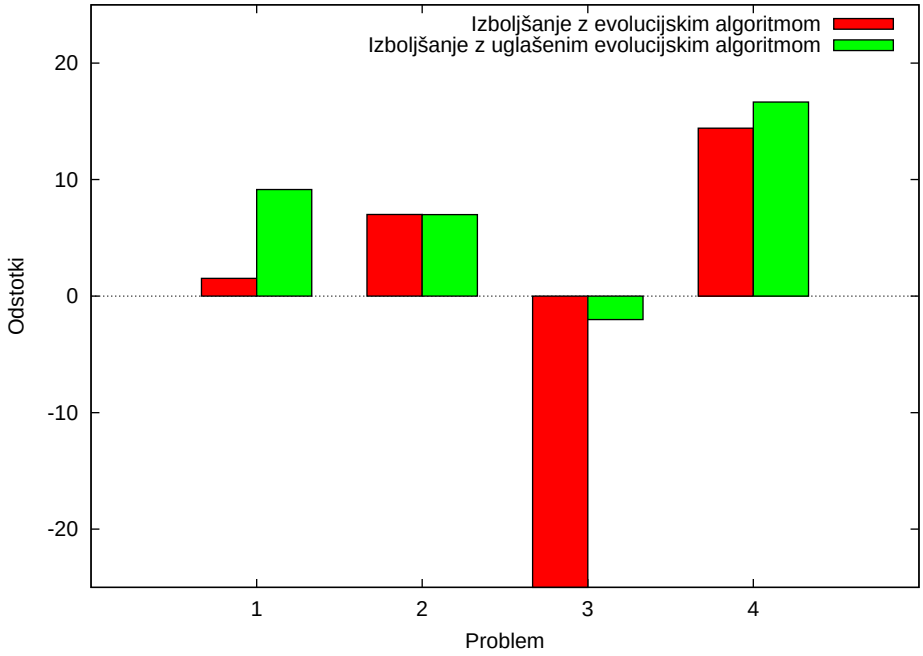
Tabela 5.11: Primerjava rezultatov.

Pri prvem, drugem in četrtem problemu je evolucijski algoritem vrnil boljšo rešitev glede na požrešno metodo. Pri tretjem problemu pa je evolucijski algoritem vrnil veliko slabšo rešitev glede na požrešno metodo. Po natančni analizi rezultatov evolucijskega algoritma pri tretjem problemu smo ugotovili, da so vzroki takih rezultatov neugodne hitrosti razkladanja, prikazane v tabeli 5.7. Predvsem neugodni sta hitrosti za tretjo vrsto tovora: 10 in 5000, katerega količina je 200000. Glede na hitrosti vozil za tretjo vrsto tovora, ki so prikazane v tabeli 5.4, je nakladanje tretje vrste tovora na vozila neugodno, saj razkladanje te vrste tovora vzame veliko časa. Kljub ocenitveni funkciji, ki oceni rešitve, katerih vozila prepeljejo približno enake odstotke vseh vrst tovora v eni vožnji, z višjo oceno, se izkaže, da pri tem problemu prevlada čas vožnje, ki je tudi del ocene. Ko v eni vožnji vozimo približno enake odstotke vseh vrst tovora, je čas vožnje dolg, saj razkladanje tretje vrste tovora vzame veliko časa. Boljšo oceno dobijo rešitve, v katerih vozila vozijo samo prvi dve vrsti tovora, kljub temu, da je odstotek prevožene tretje vrste tovora enak 0. Z analizo voženj smo ugotovili, da v začetnih vožnjah vozila vozijo le prvi dve vrsti tovora in šele ko zmanjka prvih dveh vrst tovora, začnejo voziti tretjo vrsto tovora. Tak razpored vrst tovora po vožnjah je neugoden, čemur smo se hoteli izogniti z ocenitveno funkcijo, a pri tako specifičnem problemu takšna ocenitvena funkcija ne zadošča. Algoritem bi bilo potrebno spremeniti tako, da bi optimiral celoten prevoz tovora, ne pa vsak cikel posebej. Le tako bi dosegli približno enake odstotke vseh vrst tovora v vsaki vožnji, s čimer bi se rešitve izboljšale.

V razdelku 5.3.4 smo opisali rezultate uglašenega algoritma z metaevolucijskim algoritmom. V tabeli 5.11 so prikazani rezultati, dobljeni z uglašenim evolucijskim algoritmom, in rezultati, dobljeni s požrešno metodo. Na sliki 5.2 so prikazane izboljšave rezultatov požrešnega algoritma z obravnavanim in z

uglašenim algoritmom v odstotkih. Na sliki 5.2 in tabeli 5.11 lahko opazimo, da smo z uglasitvijo evolucijskega algoritma izboljšali rezultate v primerjavi s požrešno metodo. Le pri drugem testnem problemu ni bilo izboljšave, saj je že osnovni algoritem našel zelo dobro rešitev.

Pri prvem testnem problemu smo z uglasitvijo izboljšali rezultat obravnavanega algoritma za 8,4%, pri tretjem testnem problemu za 652,21% in pri četrtem testnem problemu za 3,6%. Osnovni algoritem je vrnil boljše rezultate glede na požrešno metodo pri prvem, drugem in četrtem testnem problemu. Pri tretjem testnem problemu je osnovni algoritem vrnil kot rezultat 526,62 ur, požrešna metoda pa je vrnila 68,63 ur. Pri tem problemu je najbolj očitna izboljšava rezultata, ki ga dobimo z uglašenim algoritmom, ki je 70,01 ur. Problem je zelo zahteven, kot je bilo že predhodno opisano. Kljub temu pa je uglašeni algoritem vrnil le za 2% slabši rezultat od požrešne metode, kar dokazuje njegovo robustnost in uporabnost tudi za zelo zahtevne probleme.



Slika 5.2: Primerjava rezultatov požrešnega algoritma z rezultati evolucijskega in uglašenega evolucijskega algoritma.

Poglavje 6

Zaključek

V okviru diplomske naloge smo razvili evolucijski algoritem za reševanje problema prevoza tovora med dvema lokacijama s skupino vozil. Testirali smo ga na štirih testnih problemih. Algoritem je v povprečju našel boljše rešitve v primerjavi s požrešno metodo. Le pri zelo specifičnem problemu je algoritem vrnil slabše rešitve v primerjavi s požrešno metodo. Za uglasitev algoritma smo razvili metaevolucijski algoritem, ki je zelo izboljšal rezultate najtežjega testnega problema in rahlo izboljšal rezultate ostalih problemov. Algoritem najde rešitve problemov v sprejemljivem času. Zato je primeren za implementacijo v programskem orodju za optimizacijo prevoza.

Algoritem bi lahko izpopolnili na več načinov. Trenutna izvedba mutacije z določeno verjetnostjo mutira vse tri podatke o vozilu naenkrat. Namesto tega bi lahko vpeljali tri vrste mutacije, eno za vsak podatek o vozilu. Vsaka vrsta mutacije bi se izvedla z lastno verjetnostjo, ki bi jo določili z metaevolucijskim algoritmom.

Algoritmu lahko dodamo lokalno optimizacijo. To bi uporabili pri mutaciji in pri popravljanju kromosomov. Namesto naključnega izbora vrednosti podatkov o vozilu bi lahko te vrednosti določili z lokalno optimizacijo, na primer spremembo kromosoma v smeri največje izboljšave ocene.

Vrednosti parametrov algoritma bi lahko pametneje določili z njihovo optimizacijo med izvajanjem evolucijskega algoritma. Pri tem bi lahko uporabili mehanizme evolucijskih strategij oziroma evolucijskega programiranja.

Verjetnost samonakladanja je izračunana z vnaprej določeno formulo. Potrebno je preučiti boljše načine izbire verjetnosti samonakladanja. Lahko bi verjetnost samonakladanja dodali med parametre in bi se njena vrednost določala na enak način kot vrednosti ostalih parametrov.

Ocenjevanje kromosomov bi lahko izvedli tako, da bi vedno ocenjevali ce-

loten prevoz, ne samo posameznih ciklov. V ta namen bi bilo potrebno spremeniti predstavitev rešitev in zasnovo algoritma. Sprememba v tej smeri bi z veliko verjetnostjo prinesla večje izboljšave rezultatov.

Slike

Tabele

Literatura

[1] D. Ashlock, *Evolutionary Computation for Modeling and Optimization*, New York: Springer, 2005.

[2] T. Back, H. P. Schwefel, "An Overview of Evolutionary Algorithms for Parameter Optimization", *Evolutionary Computation*, št. 1, str. 1-23, 1993.

[3] W. Banzhaf, P. Nordin, R. E. Keller, F. D. Francone, *Genetic Programming – An Introduction*, San Francisco: Morgan Kaufmann, 1998.

[4] L. Davis, *Handbook of Genetic Algorithms*, New York: Van Nostrand Reinhold, 1991.

[5] E. Dovgan, B. Filipič, M. Medeot, G. Papa, B. Plestenjak, *Optimalni transport večje količine tovora med dvema lokacijama s skupino vozil, projektno poročilo*, Ljubljana: Inštitut za matematiko, fiziko in mehaniko, 2008.

[6] A. E. Eiben, Z. Michalewicz, M. Schoenauer, J. E. Smith, "Parameter Control in Evolutionary Algorithms", *Computational Intelligence*, št. 54, str. 19–46, 2007.

[7] A. E. Eiben, J. E. Smith, *Introduction to Evolutionary Computing*, Berlin: Springer, 2003.

[8] P. Kokol, Š. H. Babič, V. Podgorelec, M. Zorman, *Inteligentni sistemi*, Maribor: Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, 2001.

[9] K. F. Man, K. S. Tang, S. Kwong, *Genetic Algorithms*, London: Springer, 1999.

[10] M. Mernik, M. Črepinšek, V. Žumer, *Evolucijski algoritmi*, Maribor: Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, 2003.

[11] S. Meyer-Nieberg, H. G. Beyer, "Self-Adaptation in Evolutionary Algorithms", *Computational Intelligence*, št. 54, str. 47–75, 2007.

[12] Z. Michalewicz, M. Schmidt, "Parameter Control in Practice", *Computational Intelligence*, št. 54, str. 277–294, 2007.

[13] K. V. Price, R. M. Storn, J. A. Lampinen, *Differential Evolution*, Berlin: Springer, 2005.

Izjava

Izjavljam, da sem diplomsko nalogo izdelal samostojno pod vodstvom mentorja doc. dr. Janeza Demšarja in somentorja doc. dr. Bogdana Filipiča. Izkazano pomoč drugih sodelavcev sem v celoti navedel v zahvali.

Ljubljana, 1. 9. 2008 Erik Dovgan