**KAUNO TECHNOLOGIJOS UNIVERSITETAS**

**INFORMATIKOS FAKULTETAS**

**Modulio P170B400 „Algoritmų sudarymas ir analizė“**

Laboratorinio darbo ataskaita

**IP individuali užduotis**

|  |  |
| --- | --- |
|  | **Dėstytojas**  Lekt. Makackas Dalius |
|  |
| **Studentas**  Rokas Puzonas |

**KAUNAS, 2023**

# Asimptoninis sudėtingumas

|  |  |  |
| --- | --- | --- |
|  | Laikas | Kiekis |
| public void IterateSolution() |  |  |
| { | c1 | 1 |
| double bestScore = paths.First().Key; | c2 | 1 |
| double worstScore = paths.Last().Key; | c3 | 1 |
|  |  |  |
| var removedPaths = new List<double>(); | c4 | 1 |
| var parentPool = new List<List<int>>(); | c5 | 1 |
| foreach (var entry in paths) | c6 | n |
| { |  |  |
| double probability = (entry.Key - bestScore) / (worstScore - bestScore); | c7 | n |
| if (rand.NextSingle() >= probability) | c8 | n |
| { |  |  |
| parentPool.Add(entry.Value); | c9 | k |
| } |  |  |
| else |  |  |
| { |  |  |
| removedPaths.Add(entry.Key); | c10 | n-k |
| } |  |  |
| } |  |  |
|  |  |  |
| foreach (var key in removedPaths) | c11 | n/2 |
| { |  |  |
| paths.Remove(key); | c12 | n/2 |
| } |  |  |
|  |  |  |
| while (batchSize > paths.Count) | c13 | n/2 |
| { |  |  |
| var parentIdx = rand.Next(0, parentPool.Count); | c14 | n/2 |
| var parent = parentPool[parentIdx]; | c15 | n/2 |
| var child = new List<int>(parent); | c16 | n/2 |
| MutatePath(child); | c17 | n/2 |
| var cost = GetPathCost(child, places); | c18 | n/2 |
| if (!paths.ContainsKey(cost)) | c19 | n/2 |
| { |  |  |
| paths.Add(cost, child); | c20 | n/2 |
| } |  |  |
| } |  |  |
|  |  |  |
| iteration++; | c21 | 1 |
| } |  |  |

Vidutinis atvėjis:

# Programos abstraktus aprašas

Kiekvienos iteracijos metu:

1. Atmesti blogus kelius kurie surinko žemą taškų kiekį
2. Iš likusių kelių sukurti naujus kelius, kad užpildyti ištrintus kelius
3. Sukuriant naujus kelius paimnti gero kelio kopiją ir atlikti mutacijas ant kelio
4. Apskaičiuoti naujo kelio taškus ir įterpti į rikiuotą visų kelių sąrašą
5. Kartoti nuo pirmo žingsnio kol gaunamas patenkinamas rezultatas

# Galimų pasirinkimų analizė

Parametrai kuriuos galima kontroliuoti:

* Populiacijos dydis – Su kiek kelių yra dirbama vienos iteracijos metu
* Mutacija, vietovės eiliškumo keitimas – Šansas apkeisti betkurią vietovę kelį su betkuriuo kita vietove esančiu kelyje.
* Mutacija, vietovės perrinkimas – Šansas parinkti visiškai kitą vietovę renkantis iš visų esančių vietovių.
* Mutacija, lokalios vietovės perrinkimas – Šansas parinkti visiškai kitą vietovę renkantis iš „lokaliai“ esančių vietovių.
* Lokalios vietovės spindulys – Atstumas nuo vienos vietovės iki kitos kuris pasako ar tos vietovės yra „lokalios“

# Maršrutų visualizacija ir analizavimas

Po maršrutų analizavimo nustatyta, kad laikant populiacijos dydį žemai (20~50) geras maršrutas yra greičiau surandamas negu naudojant didesnius populiacijos dydžius (100~150). Kadangi tarp iteracijų išlieka didesnis skaičius vidutiniškų maršrutų.

Mutacijų šansus optimaliausia buvo laikyti apie 0.5~2%, ness su tokiomis reikšmėmis maršrute tiktais ant kelių iš vietovių įvyksta mutacija. Žemas mutacijos šansas leidžia per kelias iteracinas ištestuoti mažus pakeitimus, kad rasti geriausią sprendimą. Su dideliai mutacijos šansais 10% ir daugiau per mutaciją pasikeistų daug vietovių maršruto, tai kadangi mutacijos nenuspėjamu būdu dauguma pakeitimų būna blogesni variantai nuo originalaus maršruto.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated