Rješavanje problema trgovačkog putnika koristeći algoritam lokalnog pretraživanja 2-opt

Čilag, Martina

Undergraduate thesis / Završni rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Department of Mathematics / Sveučilište Josipa Jurja Strossmayera u Osijeku, Odjel za matematiku**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:126:920581

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-04-03**

Repository / Repozitorij:

Repository of School of Applied Mathematics and Informatics

Sveučilište Josipa Jurja Strossmayera u Osijeku

Odjel za matematiku

Sveučilišni preddiplomski studij Matematika i računarstvo

Martina Čilag

Rješavanje problema trgovačkog putnika koristeći algoritam lokalnog pretraživanja 2-opt

Završni rad

Osijek, 2022.

Sveučilište Josipa Jurja Strossmayera u Osijeku

Odjel za matematiku

Sveučilišni preddiplomski studij Matematika i računarstvo

Martina Čilag

Rješavanje problema trgovačkog putnika koristeći algoritam lokalnog pretraživanja 2-opt

Završni rad

Mentor:

doc. dr. sc. Domagoj Ševerdija

Komentori:

dr.sc. Mateja Đumić
dr.sc. Rebeka Čorić

Osijek, 2022.

Sažetak

U ovom završnom radu je opisan problem trgovačkog putnika. To je problem kombinatorne optimizacije u kojem je cilj pronaći najkraću rutu koja će svaki od n gradova iz zadanog skupa posjetiti točno jednom i vratiti se u početni grad. Dodatno je objašnjena heuristička metoda lokalnog pretraživanja koja se koristi za rješavanje optimizacijskih problema. Naglasak je stavljen na algoritmu lokalnog pretraživanja poznatom pod nazivom 2-opt. Izrađen je i praktični dio rada u kojem je moguće vizualizirati instance problema te njihovo rješavanje.

Ključne riječi

problem trgovačkog putnika, lokalno pretraživanje, 2-opt algoritam

Solving travelling salesman problem using local search 2-opt algorithm

Summary

This paper explains travelling salesman problem. It is a combinatorial optimization problem in which the goal is to find the shortest route that will visit each of the n cities from the given set exactly once and return to the starting city. In addition, the local search heuristic method, which is used to solve optimization problems, is also explained. For solving the travelling salesman problem, the 2-opt local search algorithm is mentioned. There is also a practical part of the paper where it is possible to visualize problem instances and their solving.

Keywords

travelling salesman problem, local search, 2-opt algorithm

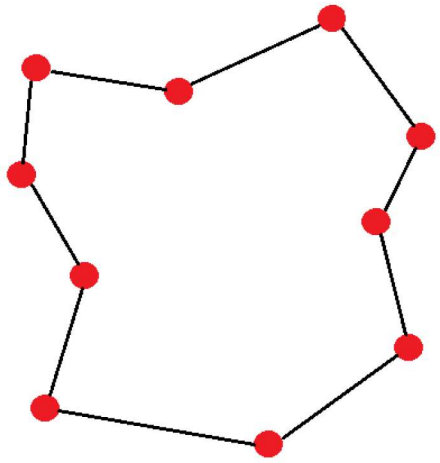
Sadržaj

1 | Uvod

Problem trgovačkog putnika problem je velike važnosti s kojim se svatko od nas susreće u svakodnevnom životu. S obzirom na moderan i ubrzan način života, svi težimo što boljem raspoređivanju vremena i što efikasnijem obavljanju dnevnih zadataka. Trebamo li posjetiti određeni broj lokacija, svatko od nas će gledati da ih posjećuje određenim redoslijedom kako ne bi imao neželjenih vraćanja koja bi samo produžila put i vrijeme. Upravo se tim problemom određivanja redoslijeda posjeta lokacija bavi problem trgovačkog putnika. U drugom će se poglavlju rada opisati problem trgovačkog putnika te će se navesti njegove inačice. Treće poglavlje se bavi algoritmima lokalnog pretraživanja, nekim njihovim prednostima i nedostacima te primjenama. U četvrtom će se poglavlju detaljnije opisati 2-opt algoritam, koji se, uz još neke algoritme, koristi za rješavanje problema trgovačkog putnika te će se navesti primjer i pseudokod algoritma. U petom je poglavlju objašnjena implementacija i vizualizacija praktičnog dijela rada.

2 | Problem trgovačkog putnika

Problem trgovačkog putnika je problem pronalaženja najkraće rute koja će svaki od n gradova iz zadanog skupa posjetiti točno jednom i vratiti se u početni grad. To je jedan od najpoznatijih problema kombinatorne optimizacije. Vrijeme nastanka formulacije problema nije točno zabilježeno. Priručnik za trgovačke putnike iz 1832. spominje problem i primjere putovanja kroz Njemačku i Švicarsku, ali u priručniku problem nije matematički formuliran. Prvi put ga je matematički razmatrao 1930.-ih godina Merrill M. Flood koji je želio riješiti problem određivanja najkraće rute školskog autobusa. Najranija publikacija koja je koristila izraz "problem trgovačkog putnika" bilo je izvješće RAND Corporation-a Julije Robinson iz 1949., "O Hamiltonijevoj igri (problem trgovačkog putnika)".[10] U narednim godinama, pa sve do danas, ovim se problemom bave znanstvenici iz različitih područja: matematike, fizike, kemije, informatike i sl.

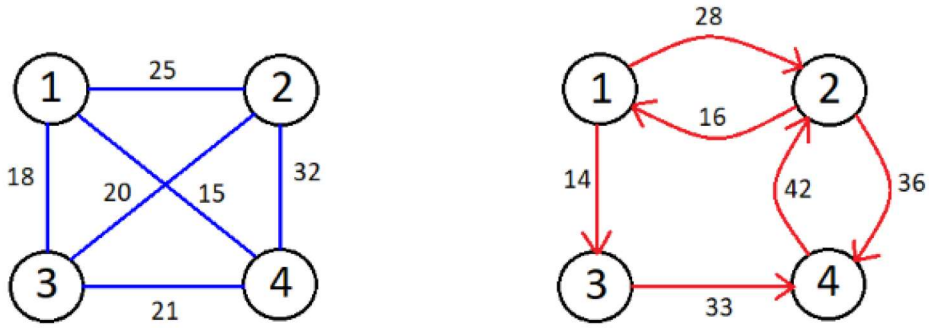


Slika 2.1: Skica rješenja problema trgovačkog putnika s 10 gradova

Kako bismo lakše mogli vizualizirati problem, on se može razmatrati kao težinski graf. Graf G = (V, E) uređeni je par čvorova (engl. *vertices*) i bridova (engl. *edges*). Težinski graf je graf u kojem je svakom bridu dodijeljena težina. Kod ovog problema čvorovi predstavljaju gradove, bridovi puteve, dok se težina odnosi na udaljenosti između gradova.

2.1 Simetričan problem trgovačkog putnika

Težinske grafove prema usmjerenosti dijelimo na usmjerene i neusmjerene. Kod usmjerenih, bridovi imaju smjer te ako između dva čvora postoje bridovi u oba smjera, njihova težina ne mora biti jednaka. Kod neusmjerenih grafova, između dva čvora postoji samo jedan brid kojim čvorove možemo posjećivati u oba smjera i kojem je pridružena samo jedna težina.



Slika 2.2: Primjer neusmjerenog težinskog grafa (lijevo) i usmjerenog (desno)

Zbog toga, probleme trgovačkog putnika dijelimo na simetrične i asimetrične. Kod simetričnih problema, udaljenosti između gradova jednake su u oba smjera te tvore neusmjeren težinski graf. Kod asimetričnih problema, udaljenosti ne moraju postojati u oba smjera, ako i postoje, mogu biti različite. Te udaljenosti tada tvore usmjeren težinski graf. U nastavku će naglasak biti stavljen na rješavanje simetričnog problema trgovačkog putnika.

Jedna od metoda rješavanja bila bi isprobavanje svih mogućih permutacija kako bismo vidjeli koja bi dala optimalno rješenje. Ta se metoda naziva potpuno pretraživanje (engl. *brute-force search*). Problem ove metode je što je njeno vrijeme izvršavanja $O(n!)$, odnosno faktorijel od broja gradova pa i za samo 20 gradova ova metoda postaje nepraktična [10]. Iz tog razloga, možemo koristiti neke heurističke algoritme. Heuristički algoritam je algoritam koji je dizajniran za rješavanje problema na brži i učinkovitiji način od određenih tradicionalnih metoda, međutim da bi to postigao, mora žrtvovati optimalnost, točnost, preciznost ili potpunost. Najčešće se koriste kada su dovoljna približna rješenja ili kao dobra osnova koja se tada nadopunjuje optimizacijskim algoritmima.[9] Neki od primjera heurističkih algoritama su: pohlepni algoritam, algoritam najbližeg susjeda, algoritam nasumičnog umetanja, 2-opt, 3-opt... Popis još nekih heurističkih algoritama i njihova vizualizacija može se vidjeti u [7].

3 | Lokalno pretraživanje

Lokalno pretraživanje heuristička je metoda koja se koristi za rješavanje optimizacijskih problema. Algoritmi lokalnog pretraživanja pokušavaju pronaći optimalno rješenje pretražujući cijeli prostor rješenja. Započinju s početnim rješenjem koje je dobiveno nasumično ili nekom drugom metodom. Zatim iterativno primjenom lokalnih promjena poboljšavaju rješenje sve dok ne dođu do nekog kriterija zaustavljanja. Kriteriji zaustavljanja mogu biti rješenje koje smatraju optimalnim, određen broj iteracija bez promjena, istek zadanog vremenskog ograničenja i dr. Usprkos tome što lokalno pretraživanje kroz iteracije poboljšava rješenje, ne jamči da će krajnje rješenje biti optimalno. Ono može ostati u lokalno optimalnom rješenju, koje može biti različito od globalnog, iz razloga što algoritam radi određene promjene dok se god rješenje poboljšava. Dakle, algoritam neće napraviti promjene koje bi trenutno pogoršale rješenje iako bi te promjene mogle dovesti do globalno optimalnog rješenja. Iz tog se razloga opisani postupak naziva lokalno pretraživanje.

Slika 3.1: Lokalno i globalno optimalno rješenje

Pseudokod općenitog algoritma lokalnog pretraživanja glasi:

Algoritam 1 Lokalno pretraživanje

1: **Procedura** LOKALNO-PRETRAŽIVANJE
2: Generiraj početno rješenje S_0
3: **Sve dok** kriterij zaustavljanja nije ispunjen **čini**
4: Pronađi $S \in ProstorRjesenja$ sa svojstvom $f(S) < f(S_0)$
5: $S_0 \leftarrow S$
6: **Vrati** S_0

Ako nismo zadovoljni s rješenjem koje nam je dao neki od algoritama lokalnog pretraživanja, možemo ga ponovno pokrenuti s drugačijim početnim uvjetima te na taj način možda dobiti bolje rješenje.

3.1 Tipovi lokalnog pretraživanja

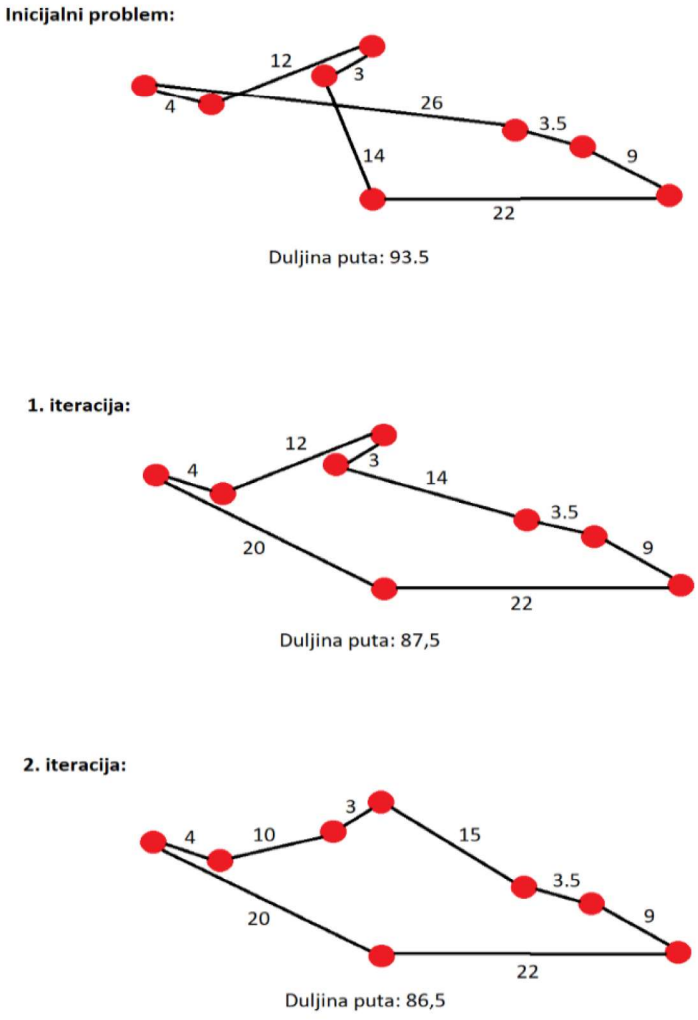
Postoje različiti tipovi lokalnog pretraživanja, neki od njih su: metoda uspona na vrh (engl. *Hill-climbing Search*), simulirano kaljenje (engl. *Simulated Annealing*), pretraživanje snopom zraka (engl. *Local Beam Search*), itd. Kod metode uspona na vrh cilj je pretraživanje brda i pronalazak najviše točke (vrha) tog brda koja predstavlja optimalno rješenje danog problema. Temelji se na tehnici heurističkog pretraživanja u kojoj osoba koja se penje na brdo procjenjuje smjer koji će ju dovesti do najvišeg vrha.[8] Ova metoda relativno brzo pronalazi rješenje, međutim ima određena ograničenja. Jedno od njih je već spomenuto lokalno optimalno rješenje. Dakle, naći će rješenje (vrh) koje je najbolje od svih susjednih, ali to rješenje ne mora uvijek biti i globalno optimalno. Isto tako mogu postojati ravna područja u kojima je teško odlučiti kojim se smjerom treba nastaviti kretati. Može se dogoditi da problem ima dva ili više lokalno optimalna rješenja pa je također teško odlučiti kada se treba zaustaviti. Simulirano kaljenje slično je metodi uspona na vrh, ali umjesto da svaki put odabere najbolji idući potez, ova metoda radi i nasumične poteze. Ako taj potez vodi do poboljšanja trenutnog rješenja, prihvatimo ga. U suprotnom, algoritam će napraviti i potez koji pogoršava trenutno stanje, ali s vjerojatnošću manjom od 1. Vjerojatnost da ćemo postići optimalno rješenje eksponencijalno se smanjuje sa svakim krivim potezom. Pretraživanje snopom zraka odabire k nasumično generiranih stanja i proširuje ih u svakom koraku. Ako je neko od stanja optimalno, metoda se zaustavlja s uspjehom. U suprotnom, odabire k najboljih nasljednika i ponavlja postupak. [8]

3.2 Primjene lokalnog pretraživanja

Algoritmi lokalnog pretraživanja imaju široku primjenu u matematici, inženjerstvu, bioinformatici te računalnoj znanosti. Navedimo neke od primjera gdje se koriste. U radu Lokalno pretraživanje u kombinatornoj optimizaciji [1] u poglavlju 6 pod nazivom Genetski algoritmi glavna je svrha proučavanje učinaka mutacije, križanja i selekcije na evoluciju genotipova u slučaju nelinearnih funkcija cilja. Ovdje se koristi računalno eksperimentiranje, između ostaloga i algoritmi lokalnog pretraživanja, u kombinaciji s teoretskom analizom. [4] Nadalje, problem minimalno povezanog dominantnog skupa vrlo je značajan problem kombinatorne optimizacije. Za njegovo rješavanje koristi se algoritam lokalnog pretraživanja s više pokretanja. Jedna od značajki tog algoritma je da može izbjeći zaustavljanje u lokalno optimalnom rješenju. [5] Još jedan primjer primjene algoritama lokalnog pretraživanja je lokacijski problem ograničenih kapaciteta. Problem se sastoji od odabira lokacija na kojima će se postaviti pogoni, skladišta i distribucijski centri. Algoritam koji rješava taj problem kombinira lokalno pretraživanje i matematičko programiranje. [3] Između ostalog, algoritmi lokalnog pretraživanja koriste se za rješavanje problema trgovačkog putnika.

4 | 2-opt algoritam

2-opt algoritam jedan je od algoritama koji koristi metodu uspona na vrh. Njegova generalizacija je k-opt algoritam. Glavna ideja k-opt algoritma je zamjena k bridova kako bismo došli do optimalnog rješenja. Svaka iteracija u tom algoritmu zahtjeva $O(n^k)$ vremena. Dakle, 2-opt algoritam razmatra sve moguće zamjene dva brida te primjenjuje onu zamjenu koja rezultira najvećim poboljšanjem rješenja. Njegovo je vrijeme izvršavanja $O(n^2)$ po iteraciji. Taj je algoritam prvi razmatrao Flood 1956., a formulirao ga je Croes 1958. godine [11].



Slika 4.1: Primjer rada 2-opt algoritma

Primjenjuje se na problem trgovačkog putnika, ali i mnoge druge povezane probleme. Što se tiče rješavanja problema trgovačkog putnika koristeći ovaj algoritam, on se može pokrenuti nakon nekog algoritma koji je već odredio rutu, kao što je, na primjer, algoritam najbližeg susjeda, kako bi ju pokušao poboljšati. Može se pokrenuti i na nasumično generiranoj ruti. U praksi ovaj algoritam daje lokalno optimalno rješenje koje je, ovisno o problemu, najčešće dosta blizu globalno optimalnog.

```
procedure 2optSwap(route, v1, v2) {
    1. take route[0] to route[v1] and add them in order to new_route
    2. take route[v1+1] to route[v2] and add them in reverse order to new_route
    3. take route[v2+1] to route[end] and add them in order to new_route
    return new_route;
}
```

Slika 4.2: Pseudokod 2-opt zamjene [11]

```
repeat until no improvement is made {
    best_distance = calculateTotalDistance(existing_route)
    start_again:
    for (i = 0; i <= number of nodes eligible to be swapped - 1; i++) {
        for (j = i + 1; j <= number of nodes eligible to be swapped; j++) {
            new_route = 2optSwap(existing_route, i, j)
            new_distance = calculateTotalDistance(new_route)
            if (new_distance < best_distance) {
                existing_route = new_route
                best_distance = new_distance
                goto start_again
            }
        }
    }
}
```

Slika 4.3: Pseudokod 2-opt algoritma [11]

Moguće je napraviti i efikasniju implementaciju algoritma. Budući da 2-opt uklanja dva brida i dodaje dva nova, možemo samo pogledati je li zbroj duljina postojeća dva brida veći od zbroja duljina nova dva. Ako je, napravimo 2-opt zamjenu. Dakle, ne trebamo raditi novu rutu i računati njenu duljinu u svakoj iteraciji.

4.1 2-opt algoritam za Euklidski problem trgovačkog putnika

Euklidski problem trgovačkog putnika koristi se u slučajevima iz stvarnog života jer su u njemu točke, odnosno koordinate, iz skupa $\mathbb{R}^2$, a za izračun udaljenosti koristi se Euklidska mjera. Njega također možemo riješiti koristeći 2-opt algoritam. Unatoč jednostavnosti algoritma, nije poznat njegov točan aproksimacijski omjer za Euklidski problem trgovačkog putnika. Godine 1999. dokazano je da je donja granica aproksimacijskog omjera $c \cdot \frac{\log n}{\log \log n}$ za neku konstantu $c > 0$, a gornja $O(\log n)$. Dakle, imamo prazninu od $O(\log \log n)$ između poznate gornje i donje granice za problem od n gradova. [2]

5 | Implementacija i vizualizacija

Projektni dio ovog rada obuhvaća implementaciju i vizualizaciju rješavanja problema trgovačkog putnika koristeći algoritme najbližeg susjeda, pohlepnog pretraživanja te 2-opt. Naglasak će biti stavljen na implementaciju i vizualizaciju 2-opt algoritma. Za implementaciju korišten je programski jezik Python, dok je za vizualizaciju korišten višeplatformski skup Python modula Pygame.

5.1 Instance problema

TSPLIB je biblioteka koja sadrži primjere instanci za problem trgovačkog putnika i povezane probleme iz različitih izvora. Više o njoj može se vidjeti u [6]. Odabrano je 6 instanci za koje implementirani algoritmi rješavaju problem: eil51, eil76, eil101, kroA100, kroC100, kroD100. Svaki od tih problema dodatno sadrži i datoteku s optimalnim rješenjem koje je matematički dokazano. Ono korisniku omogućava da usporedi rješenje dobiveno nekim od algoritama s optimalnim rješenjem. Uz već navedene probleme, dodatno je moguće generirati problem od 25 gradova s nasumično generiranim koordinatama kako bi se na relativno malom problemu lakše mogao razumjeti rad algoritama.

5.2 Računanje udaljenosti

Jednostavnim regularnim izrazom (engl. *regex*) iz instanci problema uzimamo gradove i pripadne koordinate te ih spremamo u rječnik pomoću kojeg zatim računamo međusobne udaljenosti između svaka dva grada. To možemo napraviti na dva načina. Pomoću L_1 i L_2 mjere. L_1 mjera je Manhattan mjera i računamo ju na sljedeći način:

$$d_1((x_1, y_1), (x_2, y_2)) = |x_1 - x_2| + |y_1 - y_2|$$

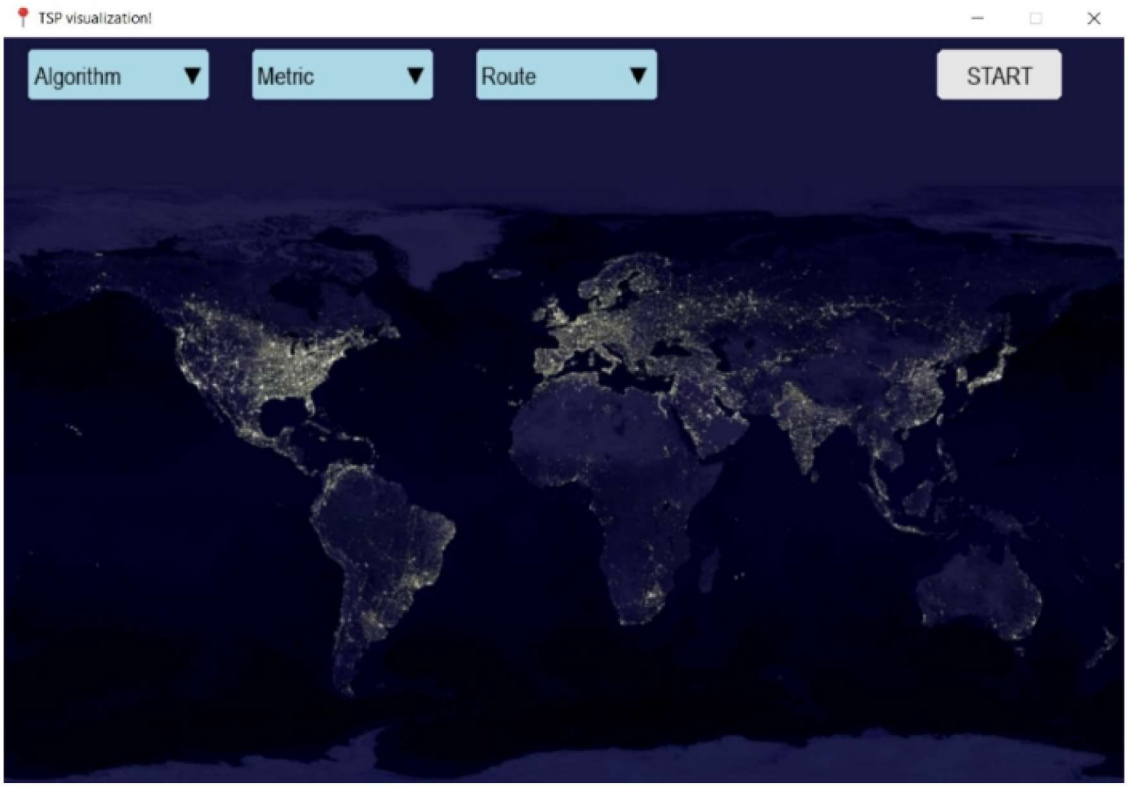
Euklidsku, odnosno L_2 mjeru, računamo na sljedeći način:

$$d_2((x_1, y_1), (x_2, y_2)) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

5.3 Implementacija 2-opt algoritma

Sama implementacija napravljena je po uzoru na pseudokod sa Slike 4.3. Prije samog pokretanja 2-opt algoritma, prvo se pokrene algoritam najbližeg susjeda kako bi se napravila početna ruta koju zatim poboljšavamo ovim algoritmom. Algoritam najbližeg susjeda kreće od nekog grada te ga spaja s najbližim sljedećim. Ponavlja taj postupak dok nisu spojeni svi gradovi. Naposlijetku zadnji grad spoji s početnim. Dakle, 2-opt uzima rutu dobivenu algoritmom najbližeg susjeda te dok se god rade promjene bridova, on iterira po ruti uklanjajući dva brida i dodajući dva nova kojima se smanjuje ukupna udaljenost rute.

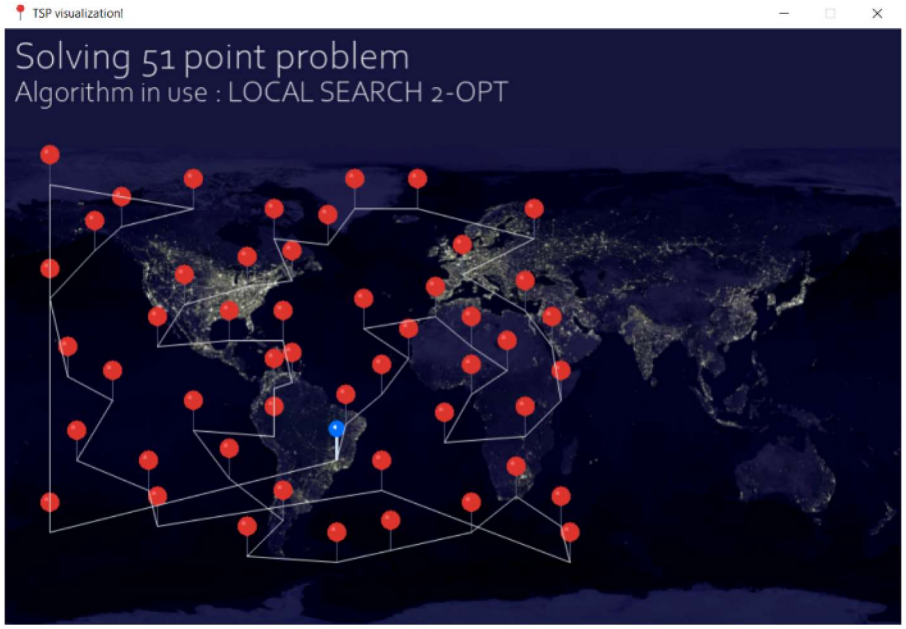
5.4 Korisničko sučelje



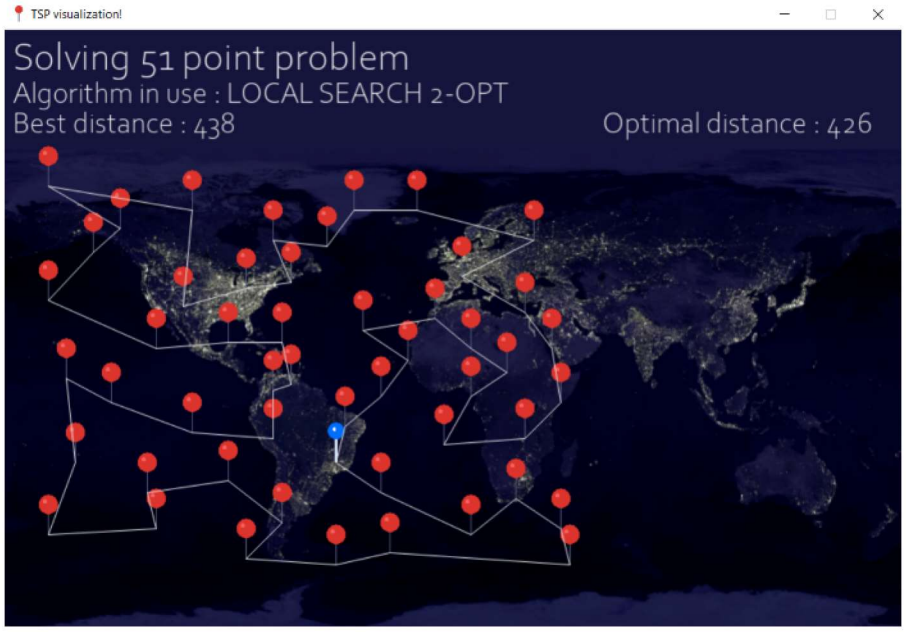
Slika 5.1: Korisničko sučelje

Korisnik treba odabrati koja od tri implementirana algoritma želi koristiti za rješavanje problema, zatim želi li koristiti L_1 ili L_2 mjeru za izračun udaljenosti te na kraju treba odabrati instancu problema. Sve su tri opcije prikazane u obliku padajućeg izbornika. Pristikom na gumb START započinje rješavanje i vizualizacija odabranog problema.

5.5 Primjer vizualizacije 2-opt algoritma



Slika 5.2: Inicijalna ruta



Slika 5.3: Ruta nakon djelovanja 2-opt algoritma

Za ovaj primjer odabran je 2-opt algoritam, L_2 mjera te instanca problema eil51. Na Slici 5.2 prikazana je inicijalna ruta koja je dobivena algoritmom najbližeg susjeda, dok je na Slici 5.3 prikazana ruta koju dobijemo nakon što 2-opt algoritam

završi. Dogodilo se ukupno 14 zamjena bridova. Nakon što algoritmi završe, na sučelju se pojavljuju udaljenosti dobivene odabranim algoritmima te optimalne udaljenosti. Udaljenost koju dobijemo pokretanjem algoritma najbližeg susjeda na ovom problemu iznosi 511, dok se pokretanjem 2-opt algoritma dobije udaljenost 438. Optimalna je udaljenost 426. Dakle, rješenje dobiveno 2-opt algoritmom vrlo je blizu optimalnog.

5.6 Spremanje rješenja

Kako korisnik ne bi ostao samo s vizualnim prikazom rješenja, svaki put kada se pokrene određeni problem, automatski se svi podaci vezani za taj problem i za samo rješenje problema spremaju u tekstualnu datoteku *Route.txt*. Ona sadrži sljedeće podatke o problemu: odabranu rutu, udaljenost dobivenu odabranim algoritmom, optimalnu udaljenost, odabrani algoritam, odabranu mjeru, broj gradova, koordinate gradova te rutu kojom treba posjećivati gradove kako bi postigli dobivenu udaljenost.

Literatura

[1] Aarts, Emile, and Jan Karel Lenstra, editors., *Local Search in Combinatorial Optimization*, Princeton University Press, 2003. JSTOR.

[2] Ulrich A. Brodowsky Pontsheide, *The Approximation Ratio of the 2-Opt Heuristic for the Euclidean Traveling Salesman Problem*, dostupno na `https://arxiv.org/pdf/2010.02583.pdf`

[3] Carolina Lagos, Guillermo Guerrero, Enrique Cabrera, Stefanie Niklander, Franklin Johnson, Fernando Paredes, Jorge Vega, *"A Matheuristic Approach Combining Local Search and Mathematical Programming"*, Scientific Programming, vol. 2016, Article ID 1506084, 7 pages, 2016.

[4] Mühlenbein, Heinz, *"Genetic Algorithms." Local Search in Combinatorial Optimization*, edited by Emile Aarts and Jan Karel Lenstra, Princeton University Press, 2003, pp. 137–72. JSTOR

[5] Li, Ruizhi, Shuli Hu, Huan Liu, Ruiting Li, Dantong Ouyang, and Minghao Yin. 2019. *"Multi-Start Local Search Algorithm for the Minimum Connected Dominating Set Problems"* Mathematics 7, no. 12

[6] Ruprecht-Karls-Universität Heidelberg, *TSPLIB*, dostupno na `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95`.

[7] STEM Lounge, *11 Animated Algorithms for the Traveling Salesman Problem*, dostupno na `https://stemlounge.com/animated-algorithms-for-the-traveling-salesman-problem` .

[8] Tutorial And Example, *Hill Climbing Algorithm in AI*, dostupno na `https://www.tutorialandexample.com/hill-climbing-algorithm`.

[9] V. Kenny, M. Nathal, S. Saldana, *Heuristic algorithms*, dostupno na `https://optimization.mccormick.northwestern.edu/index.php/Heuristic_algorithms`.

[10] WIKIPEDIA, *Travelling salesman problem*, dostupno na `https://en.wikipedia.org/wiki/Travelling_salesman_problem`.

[11] WIKIPEDIA, *2-opt*, dostupno na `https://en.wikipedia.org/wiki/2-opt`.