Traveling Salesman Problem with Time Windows

Mihaela Filipović, Lazar Bećarević

Oktobar, 2025.

Contents

1 Uvod

Ovaj projekat upoređuje dva različita pristupa rešavanju Problema trgovačkog putnika sa vremenskim prozorima (TSPTW): egzaktno rešenje zasnovano na Brute Force-u (Branch-and-Bound) i metaheuristiku pomoću simuliranog kaljenja (SA). Cilj je oceniti njihovu efikasnost u pogledu kvaliteta rešenja, računarskog vremena i skalabilnosti. Rezultati prikazuju razliku u upotrebi egzaktnih metoda i heuristika pri rešavanju kombinatornih optimizacionih problema sa vremenskim prozorima.

Problem trgovačkog putnika (TSP) je dobro poznat NP-težak problem gde je cilj pronaći najkraću moguću rutu koja posećuje skup gradova tačno jednom i vraća se do polazne tačke. U varijanti poznatoj kao Problem trgovačkog putnika sa vremenskim prozorima (TSPTW), svaki grad mora biti posećen u okviru određenog vremenskog prozora.

Ovaj projekat implementira i upoređuje dve metode za rešavanje TSPTW:

- **Brute Force (Branch-and-Bound)**: Egzaktni rešavač koji garantuje optimalno rešenje za male instance problema.
- **Simulirano kaljenje (SA)**: Metaheuristika koja daje aproksimativna rešenja za veće instance u razumnom vremenu.

Upoređivanje se zasniva na kvalitetu rešenja, performansama u vremenu izvršenja i skalabilnosti.

2 Definicija problema

TSPTW se formalno definiše na sledeći način:

- Dat je skup gradova $\{0, 1, 2, \ldots, n-1\}$, gde je grad 0 depo (polazna tačka).
- Vreme putovanja između bilo koja dva grada predstav je matricom $T[i][j]$.
- Svaki grad i ima vremenski prozor $[e_i, l_i]$ (najraniji i najkasniji dolazak), i trajanje usluge s_i koje se mora poštovati prilikom posete.
- Cilj je pronaći minimalno ukupno vreme putovanja, uz poštovanje vremenskih prozora za svaki grad.

Matematički, želimo da minimizujemo:

$$\min_X \sum_{i=1}^{n-1} T(x_i, x_{i+1}) + T(x_n, x_1)$$

uz ograničenja:

$$e_i \leq t_i \leq l_i \quad \text{za svaki grad } i$$

gde je t_i vreme u kojem se grad i posećuje.

3 Pristupi rešavanju TSPTW

3.1 Brute Force (Branch-and-Bound)

Brute Force, odnosno Branch-and-Bound, je egzaktna metoda za rešavanje malih instanci TSPTW. Sistematski istražuje sve moguće rute, osiguravajući da se nađe optimalno rešenje. Međutim, ova metoda postaje veoma zahtevna što broj gradova raste, ograničavajući njenu primenu na relativno male instance (tipično $n \leq 20$).

Rešavač je implementiran u fajli `tsptw_bruteforce.py` i podržava:

- Heuristike redosleda čvorova (najraniji, krajnji, najbliži).
- Vremenska ograničenja za kontrolu trajanja izvršenja.
- Čuvanje rešenja u JSON formatu.

3.2 Simulirano kaljenje (SA)

Simulirano kaljenje je probablistička metaheuristika inspirisana procesom žarenja u metalurgiji. Pretražuje približno optimalno rešenje iterativnim poboljšanjem datog rešenja i ponekad prihvatanjem gorih rešenja kako bi se izbegli lokalni minimumi. Algoritam koristi temperaturni raspored kako bi smanjio verovatnoću prihvatanja gorih rešenja tokom vremena.

Rešavač je implementiran u fajlu `tsptw_sa_v2.py` i podržava:

- Prilagodljivu brzinu hlađenja α.
- Vremensko ograničenje za proces rešavanja.
- Nasumičan seed za reproduktivnost.
- Čuvanje najboljeg rešenja u JSON formatu.

4 Opis rešenja TSPTW pomoću SA

4.1 Simulirano Kaljenje (Simulated Annealing)

`sa_solve()`

- Ova funkcija implementira glavni algoritam simuliranog kaljenja. Početna ruta se može generisati koristeći greedy pristup (funkcija `greedy_feasible_insertion`), ili se može proslediti kao argument.
- Simulirano kaljenje se sastoji od iteracija u kojima se pokušavaju promene u ruti koristeći nasumične pokrete, koji se prihvataju ili odbijaju na osnovu razlike u troškovima rute i trenutnoj temperaturi.
- Temperatura se postepeno smanjuje tokom vremena, uz kontrolisane parametre, dok se ne postigne minimum ili ne istekne vreme.
- Vraća najbolje rešenje pronađeno tokom procesa, zajedno sa logovima o napretku i parametrima.

4.2 Greedy Feasible Insertion

`greedy_feasible_insertion(inst: TSPTWInstance, order: Optional[List[int]]=None, max_tries: int=200) -> List[int]`

- Ova funkcija pokušava da izgradi početnu rutu koristeći greedy pristup, tako što postepeno dodaje gradove u najbolji mogući položaj u ruti. Ruta se počinje od grada 0 i pokušava se dodati gradove u redosledu određenom parametrima ili uz korišćenje unapred definisanog redosleda gradova.
- Funkcija vraća rutu koja može biti neoptimalna ako nije moguće pronaći validno rešenje nakon nekoliko pokušaja.

4.3 Pokreti (Neighborhoods)

Simulirano kaljenje se oslanja na promene u trenutnoj ruti koje se nazivaju "pokreti". Postoje tri glavna tipa pokreta:

4.3.1 Relocate Move

`move_relocate(route: List[int], i: int, j: int) -> Tuple[List[int], int, int]`

- Ova funkcija premesti grad `i` na poziciju `j` u ruti, ukoliko `i` nije prvi ili poslednji grad.
- Rezultat je nova ruta sa promenjenim položajem grada `i`.

4.3.2 Swap Move

`move_swap(route: List[int], i: int, j: int) -> Tuple[List[int], int, int]`

- Ova funkcija menja mesta dva grada `i` i `j` u ruti, ukoliko ni jedan od njih nije prvi ili poslednji grad.
- Rezultat je nova ruta sa zamenjenim gradovima.

4.3.3 Two-opt Move

`move_two_opt(route: List[int], i: int, j: int) -> Tuple[List[int], int, int]`

- Ova funkcija primenjuje `2-opt` promenu, koja obrće deo rute između gradova `i` i `j`.
- Ovaj pokret koristi se za smanjenje ukupne dužine rute.

4.3.4 Random Move

`random_move(route: List[int], rng: random.Random) -> Tuple[str, List[int], int, int]`

- Funkcija generiše nasumičan pokret, birajući između relociranja, zamene ili 2-opt pokreta.
- Na osnovu odabranog pokreta, funkcija se poziva na odgovarajuću funkciju za generisanje nove rute.

4.4 Automatsko podešavanje temperature

`auto_temperature(inst: TSPTWInstance, route: List[int], rng: random.Random, sample_moves: int=200, target_accept: float=0.75, tw_soft: bool=False, beta: float=0.0) -> float`

- Funkcija automatski podešava početnu temperaturu algoritma simuliranog kaljenja na osnovu trenutne rute i instancije problema.
- Analizira se broj promena koje bi bile prihvaćene tokom uzorka, a zatim se koristi logaritamska formula za izračunavanje optimalne početne temperature.

5 Struktura projekta

Projekat se sastoji od sledećih komponenti:

- `tsptw_common.py`: Zajedničke strukture podataka i util-funkcije.
- `tsptw_bruteforce.py`: Egzaktni solver pomoću Branch-and-Bound.
- `tsptw_sa_v2.py`: Solver pomoću simuliranog kaljenja.
- `bench_runner.py`: Okvir za benchmark poređenje solvera.
- `viz_suite.py`: Alati za vizualizaciju rezultata.
- `tsptw_edge_suite/`: Test instance s graničnim slučajevima.
- `tsptw_SA_suite/`: Test instance različitih težina (lako, srednje, teško).

6 Benchmarking i vizualizacija

Da bismo ocenili performanse oba solvera, koristimo okvir za benchmarking `bench_runner.py`. Rezultati se čuvaju u `tsptw_bench_results.csv`, koji uključuje kvalitet rešenja (gap), vreme izvršenja i druge metrike. Vizualizacije se generišu pomoću `viz_suite.py`.

6.1 Rezultati benchmarka

Benchmark daje dve ključne metrike:

- **Gap (kvalitet rešenja)**: Mera razlike između SA rešenja i optimalnog rešenja:

$$\text{gap} = \frac{SA\_cost - Optimal\_cost}{Optimal\_cost} \times 100\%$$

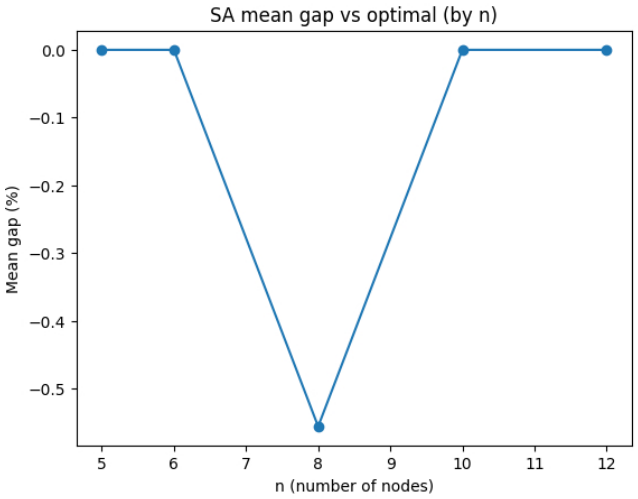
- **Speedup**: Ubrzanje Simuliranog kaljenja u odnosu na brute force metodu:

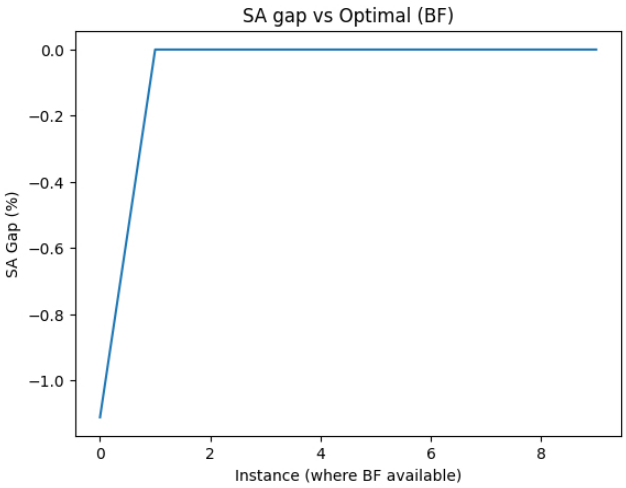
$$\text{speedup} = \frac{BF\_runtime}{SA\_runtime}$$

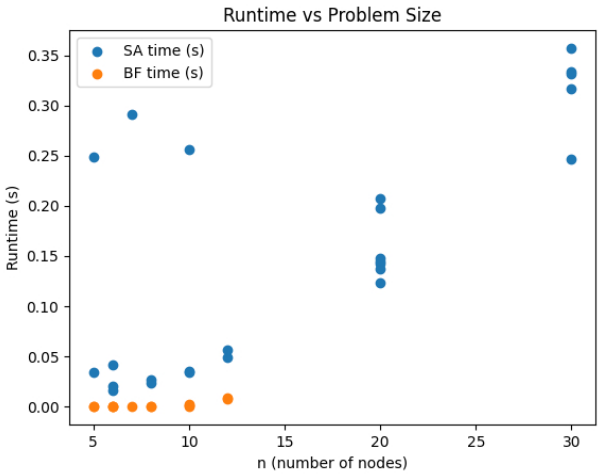
6.2 Vizualizacije

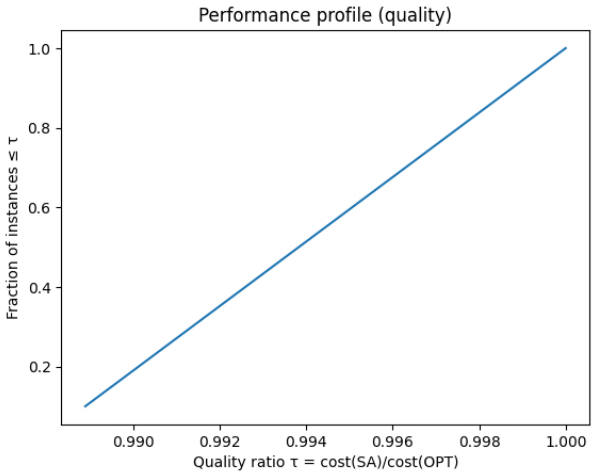
Mogu se generisati razne vizualizacije, kao što su:

- Gap u zavisnosti od veličine problema (`gap_by_n`).
- Distribucija gap-ova po nivou težine (`gap_box`).
- Poređenje vremena izvršenja (`runtime`).
- Performansni profil (`profile`).









7 Zaključak

Ovaj projekat upoređuje dva različita pristupa rešavanju Problema trgovačkog putnika sa vremenskim prozorima: Brute Force (egzaktni) i simulirano kaljenje (metaheuristika). Rezultati pokazuju da je simulirano kaljenje znatno brže i skalabilnije za veće instance, dok brute force pruža egzaktna rešenja za manje dimenzije problema. Benchmark i vizualizacije pružaju uvide u kompromise između kvaliteta rešenja i vremena izvršenja za obe metode.

8 Reference

1. Formulacija TSPTW problema: klasična literatura iz operacionih istraživanja.
2. Simulirano kaljenje: Kirkpatrick i saradnici (1983), Optimization by Simulated Annealing, Science.