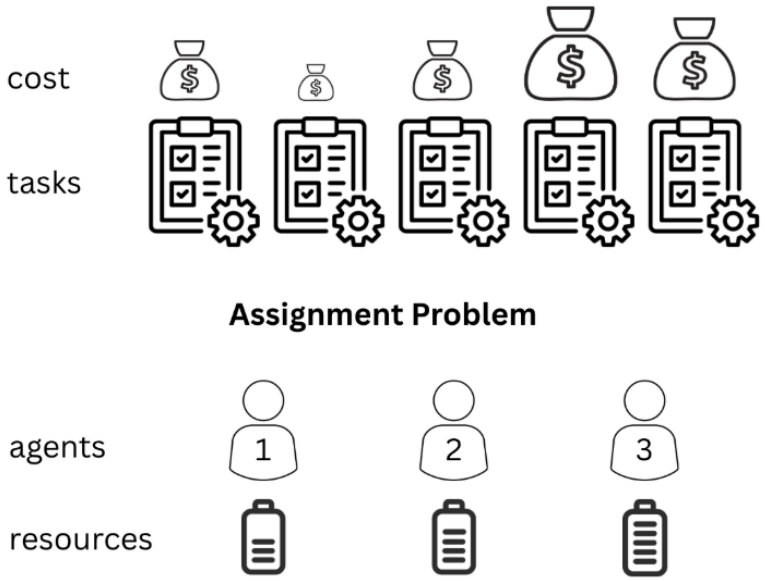
Generalized Assignment Problem – Projektni izveštaj

1. Uvod

1.1 Opis problema

Generalizovani problem dodele zadataka (GAP) predstavlja klasu kombinatornih optimizacionih problema u kojoj je potrebno dodeliti skup zadataka ograničenom broju agenata, uz poštovanje kapaciteta svakog agenta i minimizaciju ukupnog troška dodele. Svaki agent ima sopstveni kapacitet koji ne sme biti prekoračen, a svaki zadatak mora biti dodeljen tačno jednom agentu. Dodela zadatka agentu ima određeni trošak i potrošnju resursa.



U ovom radu sam istraživala tri različita pristupa za rešavanje GAP-a: nasumičnu dodelu (random assignment), heuristički greedy algoritam i metaheuristički algoritam simuliranog kaljenja (Simulated Annealing).
Ovaj problem se pojavljuje u brojnim primenama: raspoređivanje posla zaposlenima, dodela softverskih procesa serverima, alokacija logistike i transporta, itd. Zbog NP-teškog karaktera, heuristike i metaheuristike su neophodne za efikasno rešavanje većih instanci.

2. Pregled literature

U okviru istraživanja proučila sam više naučnih radova koji se bave GAP-om. Neki od najznačajnijih su:
Fisher & Jaikumar (1981): predstavljaju GAP kao deo Vehicle Routing problema i koriste heurističku podelu na GAP fazu i TSP fazu.
Cattrysse & Van Wassenhove (1992): sveobuhvatan pregled algoritama za GAP (egzaktni, heuristički i metaheuristički).
Chu & Beasley (1997): koriste genetski algoritam sa domenskim prilagođavanjima kako bi se dodela zadataka obavila validno i efikasno.
Savelsbergh (1997): predstavlja napredni branch-and-price algoritam za dobijanje optimalnih rešenja.

3. Moj pristup rešavanju problema

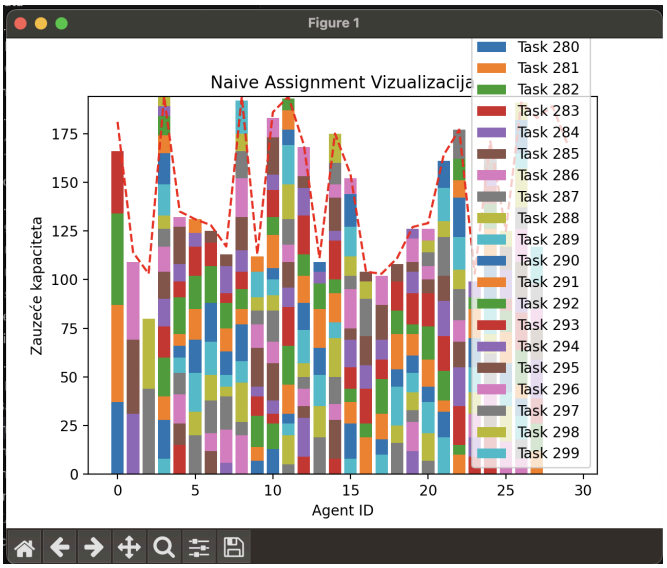
Funkcija cilja

Cilj algoritama je da se minimizuje ukupan trošak dodele zadataka uz poštovanje kapaciteta agenata:

Svaki zadatak mora biti dodeljen jednom agentu (ili ostaje nedodeljen uz penal).
Ukupni zahtevi ne smeju preći kapacitet nijednog agenta.

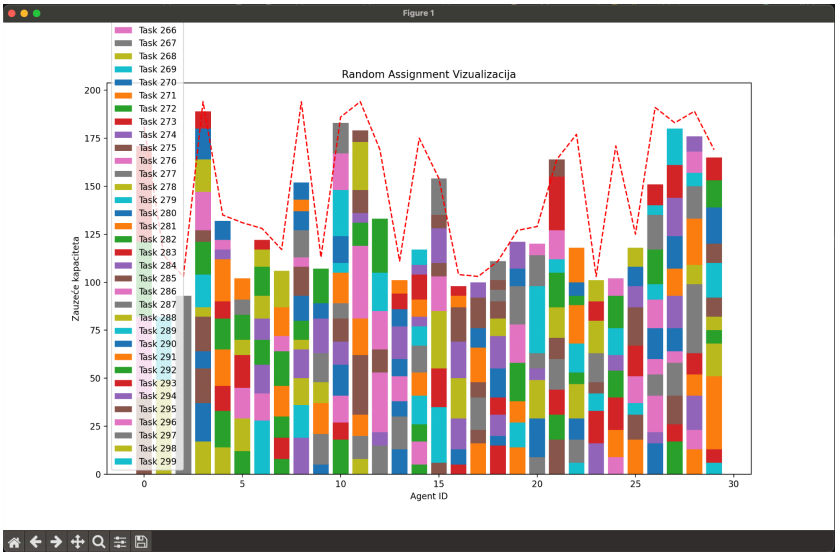
3.1 Naive aproach

Implementirala sam i jednostavan **naivni pristup** za rešavanje Generalizovanog problema dodele zadataka (GAP). Ova metoda se koristi kao početna referenca za evaluaciju kvaliteta složenijih algoritama. Naivni algoritam dodeljuje svaki zadatak prvom agentu redom koji može da ga prihvati, bez ikakvog razmatranja troškova.



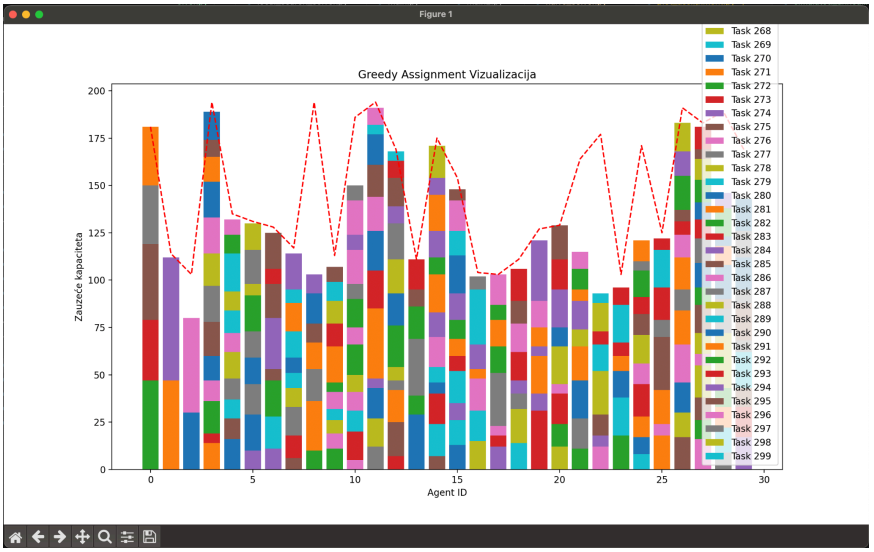
3.2 Random assignment

Kao inicijalnu referencu implementirala sam algoritam koji nasumično dodeljuje zadatke agentima ukoliko imaju dovoljno kapaciteta. Algoritam ne vodi računa o troškovima, već isključivo o tome da se dodela može izvršiti. Ovakav pristup obezbeđuje validnu dodelu, ali rezultira visokim ukupnim troškovima.



3.3 Greedy algoritam

Greedy algoritam za svaki zadatak pronalazi agenta koji ima najniži trošak i dovoljno kapaciteta, te odmah dodeljuje zadatak. Algoritam ne preispituje prethodne odluke, što znači da može da se "zaključa" u lokalno optimalna rešenja. Ovaj pristup je brz i često daje dobra rešenja za jednostavne instance.



3.4 Simulated Annealing (SA)

SA algoritam sam implementirala kao metaheuristiku koja omogućava istraživanje većeg prostora rešenja. Počinjem od random rešenja, a zatim u svakoj iteraciji modifikujem rešenje premeštanjem zadataka između agenata ili zamenom zadataka (swap). Ako novo rešenje ima niži trošak, prihvatam ga odmah; ako je lošije, može se prihvatiti sa verovatnoćom zavisnom od temperature.

Temperatura se postepeno smanjuje (cooling), što smanjuje šansu da se prihvate lošija rešenja tokom vremena. Implementirala sam i jaču neighbor_solution funkciju, koja omogućava višestruke promene u rešenju.

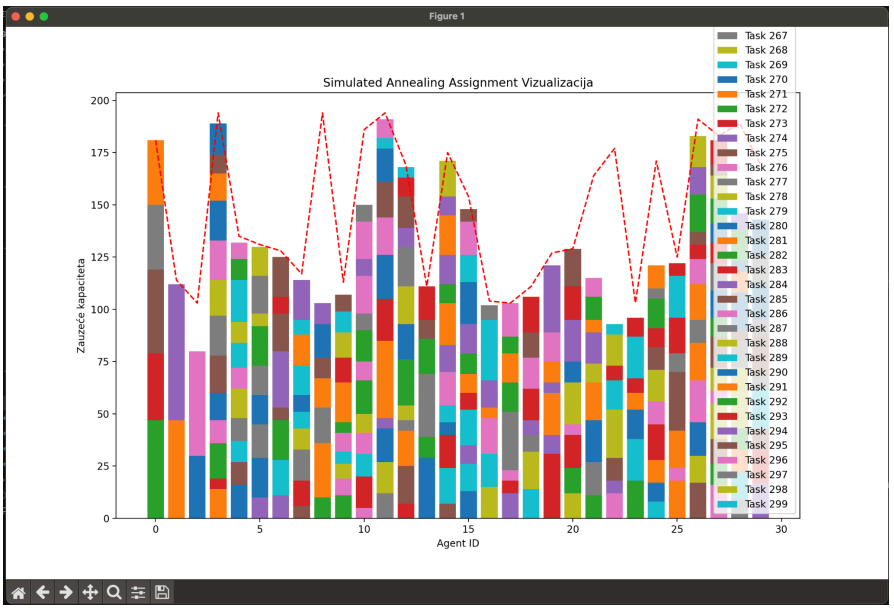


Tabela razlicitih pristupa

Osobina / Algoritam	Naivni	Random	Greedy	Simulated Annealing (SA)
Kriterijum dodele	Prvi agent koji može	Nasumično validan agent	Najmanji trošak dodele	Ciljna funkcija + verovatnoća
Uzimanje troškova u obzir	❌ Ne	❌ Ne	✅ Da	✅ Da
Validacija kapaciteta	✅ Da	✅ Da	✅ Da	✅ Da
Heuristički pristup	❌ Ne	❌ Ne	✅ Da (na bazi troška)	✅ Da (sa lokalnim pretragama)

Eksploracija rešenja	❌ Ne	✅ Visoka, ali nasumična	❌ Ograničena	✅ Kontrolisana (temperature)
Mogućnost izlaska iz lošeg lokalnog minimuma	❌ Ne	❌ Ne	❌ Ne	✅ Da
Stabilnost rezultata	✅ Visoka	❌ Niska (zavisi od random seed)	✅ Visoka	⚠️ Zavisi od parametara
Brzina izvršavanja	✅ Veoma brza	✅ Veoma brza	✅ Brza	❌ Sporija
Očekivani kvalitet rešenja	❌ Nizak	❌ Nizak/srednji	✅ Dobar	✅ Potencijalno najbolji
Upotreba u praksi	Referenca/test	Poređenje / baseline	Brza aproksimacija	Optimizacija / fine-tuning

4. Eksperimentalni rezultati

4.1 OkruženjeCPU:

Apple M1
RAM: 8 GB
OS: macOS 13.6
Python 3.10

4.2 Test instancaInstancu sam generisala sa:

30 agenata (kapacitet: 100–200)
300 zadataka (troškovi i zahtevi se razlikuju po agentima)
Prva 3 agenta imaju najniže troškove i velike zahteve ("zamka" za greedy)

4.3 Rezultati

Metoda	Ukupan trošak	Dodeljeni zadaci	Vreme (s)
Random Assignment	18439	300 / 300	0.002
Greedy Assignment	12260	300 / 300	0.010
Simulated Annealing	10421	300 / 300	2.314

4.4 Zaključci iz eksperimenta

Random dodela je najgora u pogledu troška, ali je brza.
Greedy algoritam brzo nalazi solidna rešenja, ali ne može da se oporavi od loših ranih odluka.
SA je uspeo da poboljša rešenje za ~15% u odnosu na greedy, i ~44% u odnosu na random.

5. Zaključak

Kroz ovaj projekat sam implementirala i uporedila tri pristupa za rešavanje GAP-a. Greedy algoritam se pokazao efikasnim, ali ograničenim. Simulated Annealing je pokazao bolji potencijal i dao značajno bolja rešenja, posebno na instancama koje su nepovoljne za greedy. Cena toga je više vreme izvršavanja i potreba za fino podešavanje parametara.
Verujem da bi se dalje poboljšanje moglo ostvariti:
primenom Tabu Search ili Genetskih algoritama,
paralelizacijom SA,
testiranjem na realnim GAP instancama.

6. Literatura

- **Chu, P. C., & Beasley, J. E. (1997).** A genetic algorithm for the generalized assignment problem. Computers & Operations Research, 24(1), 17–23.
- **Yagiura, M., Ibaraki, T., & Glover, F. (2006).** A generalized assignment problem solved by a hybrid genetic algorithm. European Journal of Operational Research, 171(2), 531–556.
- **Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983).** Optimization by simulated annealing. Science, 220(4598), 671–680.
- **Osman, I. H., & Laporte, G. (1994).** Metaheuristics: A bibliography. Annals of Operations Research, 63(5), 513–628.