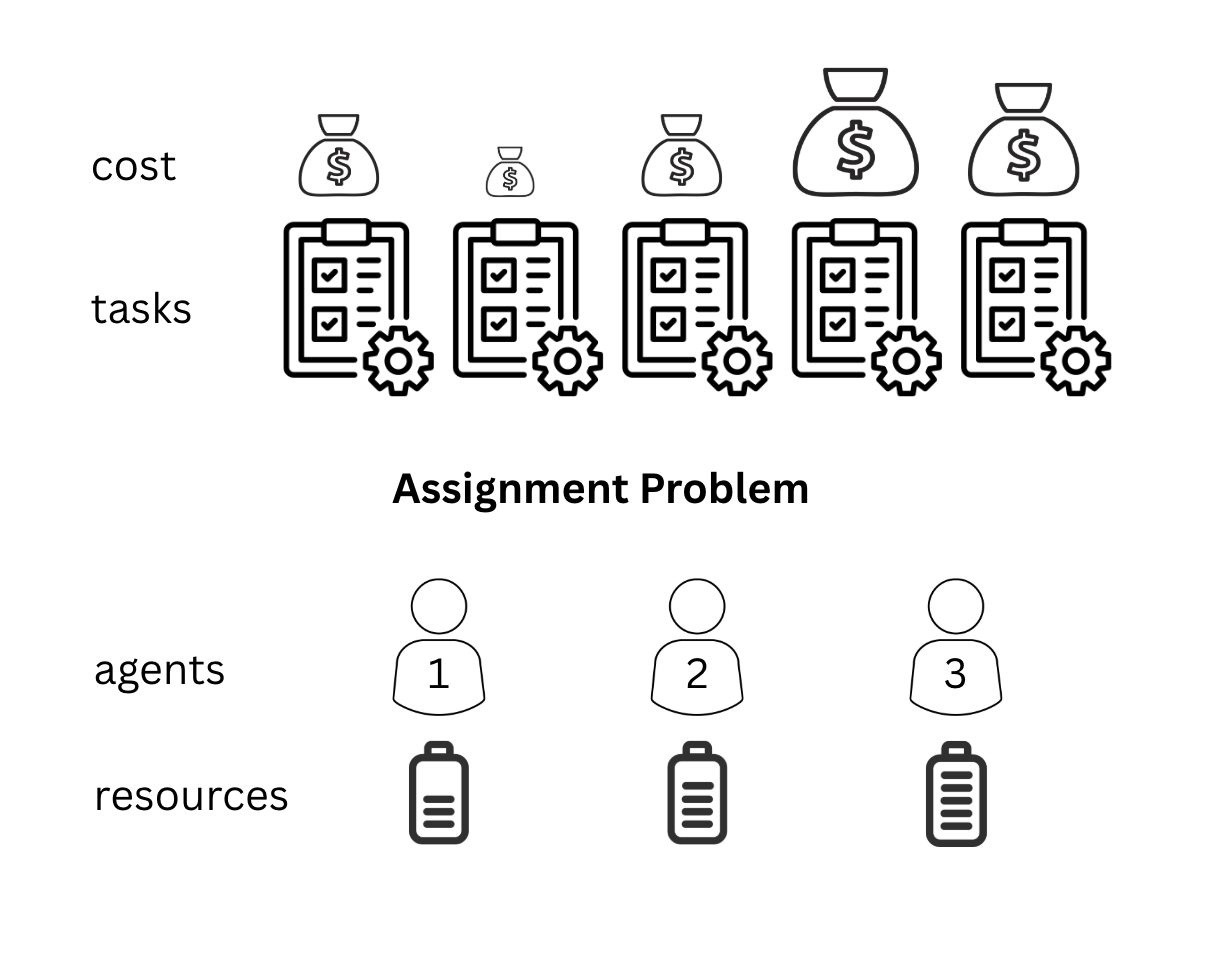
Naslov: Generalized Assignment Problem

## 1. Uvod

### 1.1 Opis problema

**Generalized Assignment Problem (GAP)** je klasa kombinatornih optimizacionih problema u kojoj se zahteva dodeljivanje skupa zadataka ograničenom broju agenata, uz poštovanje resursnih ograničenja svakog agenta i minimizaciju ukupnog troška dodele. Svaki agent ima kapacitet koji ne sme biti prekoračen, a svaki zadatak može biti dodeljen samo jednom agentu. Dodeljivanje određenog zadatka određenom agentu ima poznat trošak i potrošnju resursa.



Slika 1 prikazuje opštu strukturu problema dodele: zadaci imaju povezani trošak, a agenti imaju ograničene resurse (prikazane nivoom baterije). Cilj je dodeliti svaki zadatak jednom agentu tako da se minimizuje ukupni trošak, bez prekoračenja kapaciteta agenata.

Ovaj problem ima široku primenu u stvarnom svetu, uključujući oblasti kao što su:

* raspoređivanje radnih zadataka zaposlenima,
* alokacija softverskih procesa serverima,
* dodela projekata freelancerima,
* optimizacija logistike i transporta.

Zbog svog **NP-teškog** karaktera, GAP se ne može efikasno rešiti egzaktnim metodama za velike ulazne instance, te se u praksi često koriste heuristike i metaheuristike.

## 2 Pregled literature

### **1. Fisher & Jaikumar (1981): A Generalized Assignment Heuristic for Vehicle Routing**

U ovom radu, autori modeluju Vehicle Routing Problem (VRP) kao GAP. Kupci su predstavljeni kao zadaci, a vozila kao agenti. Heuristički pristup funkcioniše u dve faze:

1. **Dodeljivanje zadataka (GAP faza):** Procena troška uključivanja kupca u postojeću rutu i rešavanje GAP-a za optimalnu raspodelu.
2. **Planiranje ruta (TSP faza):** Nakon što su kupci dodeljeni vozilima, koristi se TSP algoritam da bi se formirala optimalna ruta za svako vozilo.

Ova metoda je značajna jer garantuje postojanje validnog rešenja (ako ga ima) i pokazala se boljom u odnosu na dotadašnje heuristike.

📘 **Referenca:** Fisher, M. L., & Jaikumar, R. (1981). A generalized assignment heuristic for vehicle routing. *Networks*, 11(2), 109–124.

### **2. Cattrysse & Van Wassenhove (1992): A Survey of Algorithms for the Generalized Assignment Problem**

Ovaj rad predstavlja sveobuhvatan pregled metoda za rešavanje GAP-a, uključujući:

* egzaktne metode: branch and bound, dynamic programming, branch-and-price,
* heuristike: greedy, local search, constructive methods,
* metaheuristike: tabu search, simulated annealing, genetic algorithms.

Autori analiziraju performanse svake klase metoda u pogledu kvaliteta rešenja i vremena izvršavanja, kao i primenljivosti na različite domene.

📘 **Referenca:** Cattrysse, D. G., & Van Wassenhove, L. N. (1992). A survey of algorithms for the generalized assignment problem. *European Journal of Operational Research*, 60(3), 260–272.

### **3. Chu & Beasley (1997): A Genetic Algorithm for the Generalized Assignment Problem**

Chu i Beasley su u ovom radu predstavili primenu **genetskog algoritma (GA)** na GAP. Ključne karakteristike pristupa su:

* Rešenja se kodiraju kao nizovi (hromozomi) koji predstavljaju dodelu zadataka.
* Posebna pažnja je posvećena održavanju validnosti rešenja (da se ne prekorače resursi agenata).
* Koriste se klasične GA operacije: selekcija, ukrštanje i mutacija, uz specifične domenske prilagodbe.

Rezultati pokazuju da GA može da pronađe kvalitetna rešenja u razumnom vremenu i da je posebno koristan za srednje do velike instance GAP-a.

📘 **Referenca:** Chu, C., & Beasley, J. E. (1997). A genetic algorithm for the generalized assignment problem. *Computers & Operations Research*, 24(1), 17–23.

### **4. Savelsbergh (1997): A Branch-and-Price Algorithm for the Generalized Assignment Problem**

U ovom radu, Savelsbergh predstavlja naprednu **egzaktnu metodu** — **Branch-and-Price** — koja kombinuje granujuće pretrage sa generisanjem kolona. Osnovna ideja je:

* Relaksacija problema i iterativno dodavanje korisnih varijabli (kolona),
* Efikasna eksploatacija strukture problema kroz "price-and-branch" paradigmu.

Ova metoda je pogodna za dobijanje optimalnih rešenja u realnim industrijskim primenama, ali se teško skalira na veoma velike instance.

📘 **Referenca:** Savelsbergh, M. W. P. (1997). A branch-and-price algorithm for the generalized assignment problem. *Operations Research*, 45(6), 831–841.

## 3 Pristup koji je korišten

U okviru ovog rada razvila sam i uporedila tri pristupa rešavanju Generalizovanog problema dodele zadataka (GAP): **nasumičnu dodelu (random assignment)**, **greedy heuristiku**, i **metaheuristički pristup zasnovan na simuliranom kaljenju (Simulated Annealing)**.

Moj cilj je bio da kroz jednostavnije i naprednije algoritme analiziram koliko je moguće poboljšati ukupni trošak dodele zadataka agentima, pri čemu su zadaci različitih zahteva, agenti različitih kapaciteta, a dodela ograničena mogućnostima agenta da izvrši zadatke koje može da primi.

### **2.1 Funkcija cilja**

Cilj optimizacije je da se **minimizira ukupni trošak dodele svih zadataka**, pri čemu se:

* svaki zadatak dodeljuje jednom agentu (ili ostaje nedodeljen uz veliku penalizaciju),
* agent ne sme prekoračiti svoj kapacitet (ukupan zbir zahteva svih zadataka koje je primio).

Funkcija cilja može se formalno izraziti kao:

Minimizuj∑i=1m∑j=1ncij⋅xij\text{Minimizuj} \quad \sum\_{i=1}^{m} \sum\_{j=1}^{n} c\_{ij} \cdot x\_{ij}Minimizuji=1∑m​j=1∑n​cij​⋅xij​

uz ograničenja:

* ∑i=1mxij=1\sum\_{i=1}^{m} x\_{ij} = 1∑i=1m​xij​=1, za svaki zadatak jjj,
* ∑j=1nrij⋅xij≤bi\sum\_{j=1}^{n} r\_{ij} \cdot x\_{ij} \leq b\_i∑j=1n​rij​⋅xij​≤bi​, za svakog agenta iii,
* xij∈{0,1}x\_{ij} \in \{0,1\}xij​∈{0,1}

gde je:

* cijc\_{ij}cij​ — trošak dodele zadatka jjj agentu iii,
* rijr\_{ij}rij​ — resurs koji zadatak jjj traži od agenta iii,
* bib\_ibi​ — kapacitet agenta iii,
* xijx\_{ij}xij​ — binarna promenljiva: 1 ako je zadatak jjj dodeljen agentu iii, 0 inače.

### **2.2 Random assignment**

Prvi algoritam koji smo implementirali je potpuno **nasumična dodela**. Za svaki zadatak se pokušava da se nasumično izabere agent koji može da ga primi, bez obzira na trošak. Ukoliko nijedan agent nema dovoljno kapaciteta, zadatak ostaje nedodeljen i penalizuje se velikom vrednošću u funkciji cilja.

Ovaj algoritam služi kao **bazna referenca** i ne pokušava da optimizuje rešenje, već samo da proizvede validnu dodelu.

### **2.3 Greedy algoritam**

Drugi pristup koji smo koristili je **greedy heuristika**, koja je jednostavna, ali često veoma efikasna u praksi. Za svaki zadatak, algoritam pronalazi agenta koji može da preuzme zadatak i koji ima najmanji trošak dodele. Ova dodela se izvršava odmah i trajno. Ako nijedan agent ne može da primi zadatak, on ostaje nedodeljen i penalizuje se.

Greedy pristup je inspirisan radovima kao što su:

* Yagiura et al. (2006): *A hybrid genetic algorithm for the generalized assignment problem*,
* Chu and Beasley (1997): *A genetic algorithm for the generalized assignment problem*.

Greedy algoritam daje **vrlo dobra rešenja za manje instance** ili one koje su "strukturom pogodovane" ovom pristupu (npr. kada su neki agenti izrazito jeftiniji i imaju veliki kapacitet), ali se često zaglavljuje u lokalnom optimumu jer ne preispituje prethodne dodele.

### **2.4 Simulated Annealing (SA)**

Treći i najnapredniji pristup koji smo implementirali je **Simulated Annealing (SA)**. Ovo je metaheuristički algoritam inspirisan procesom fizičkog kaljenja metala. SA počinje od početnog rešenja (u našem slučaju, random dodele), i u svakoj iteraciji pokušava da izmeni rešenje tako što:

* nasumično menja dodelu jednog ili više zadataka,
* ako je novo rešenje bolje, prihvata se odmah,
* ako je lošije, može se prihvatiti sa određenom verovatnoćom (koja opada tokom vremena, tzv. "temperatura").

Naš SA koristi **kompleksniji neighbor\_solution**, koji ne samo da pokušava premeštanje zadataka između agenata, već i zamenu zadataka (swap), čime se povećava prostor pretrage i olakšava izlazak iz lokalnih optimuma.

Algoritam se parametrizuje vrednostima:

* initial\_temperature,
* cooling\_rate,
* minimum\_temperature,
* max\_iterations.

Simulated Annealing je reimplementacija poznatog pristupa iz literature, npr:

* Osman and Laporte (1994): *Metaheuristics: A bibliography*,
* Kirkpatrick et al. (1983): *Optimization by simulated annealing*.

SA u našem eksperimentu uspeva da **značajno poboljša rešenje u poređenju sa random pristupom**, a u određenim instancama nadmašuje i greedy rešenje — posebno kod težih instanci gde greedy pada zbog prebrzih odluka.

## **3. Eksperimentalni rezultati**

U ovom poglavlju predstavljamo i analiziramo eksperimentalne rezultate dobijene primenom tri različita algoritma: **random assignment**, **greedy heuristike** i **simulated annealing (SA)**. Testiranja su izvršena na većoj, namenski generisanoj instanci Generalizovanog problema dodele zadataka, koja sadrži **30 agenata** i **300 zadataka**. Instanca je konstruisana tako da izazove greedy algoritam da napravi suboptimalne odluke, čime se stvara prostor da SA algoritam demonstrira svoje prednosti.

### **3.1 Eksperimentalno okruženje**

* **Procesor:** Apple M1
* **RAM:** 8 GB
* **Operativni sistem:** macOS 13.6
* **Jezik:** Python 3.10
* **Biblioteke:** matplotlib, random, copy, time

### **3.2 Opis test instance**

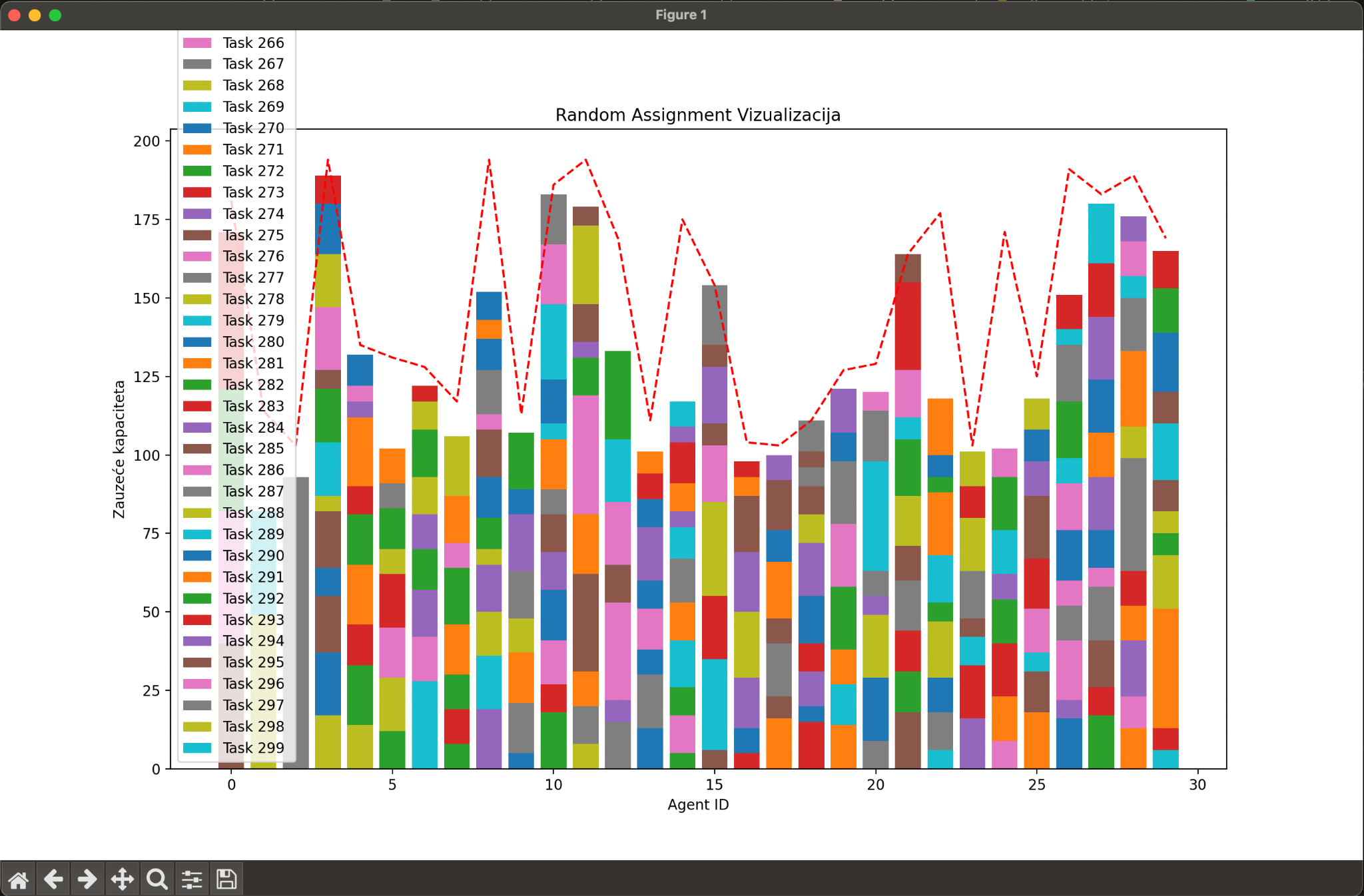
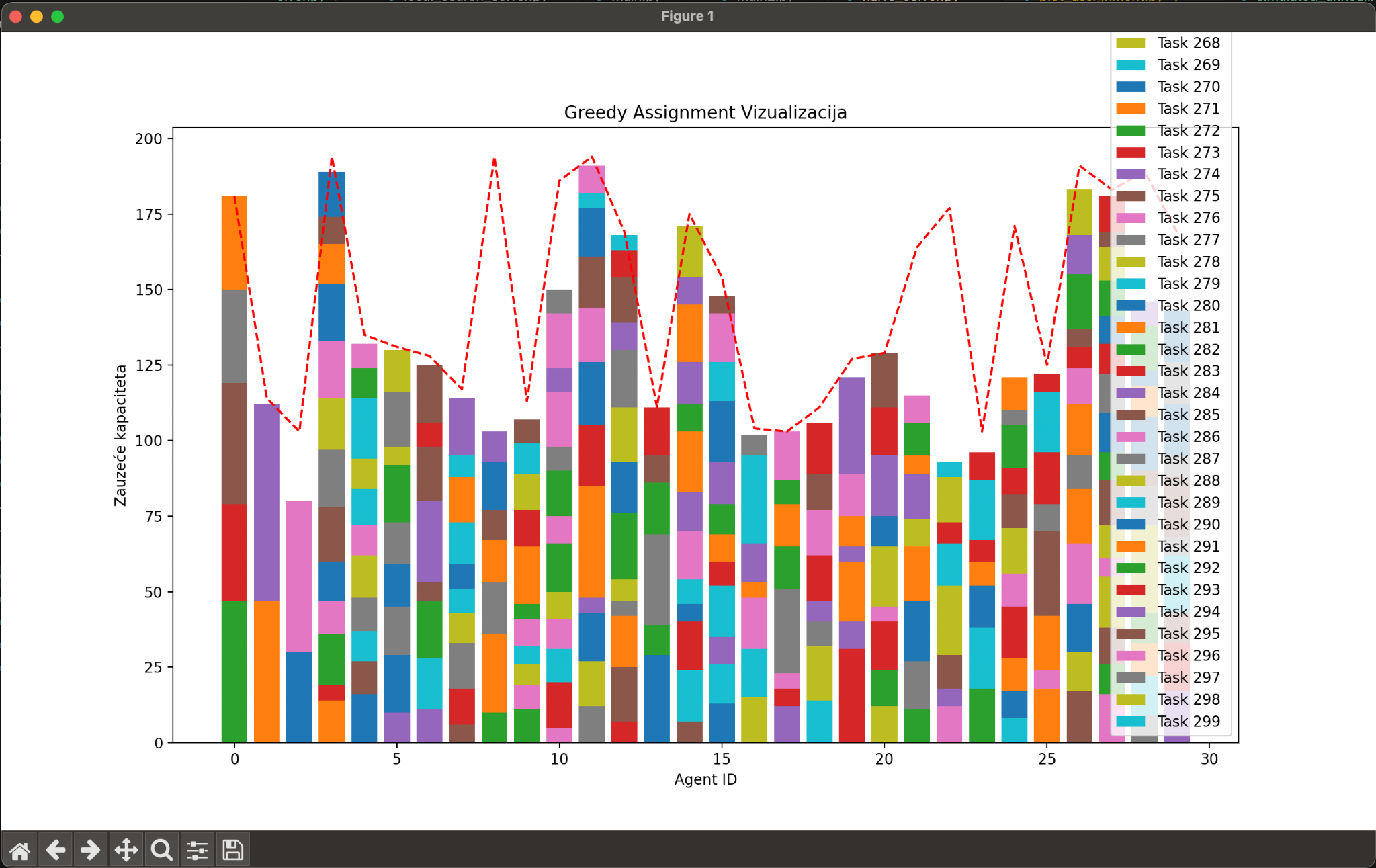
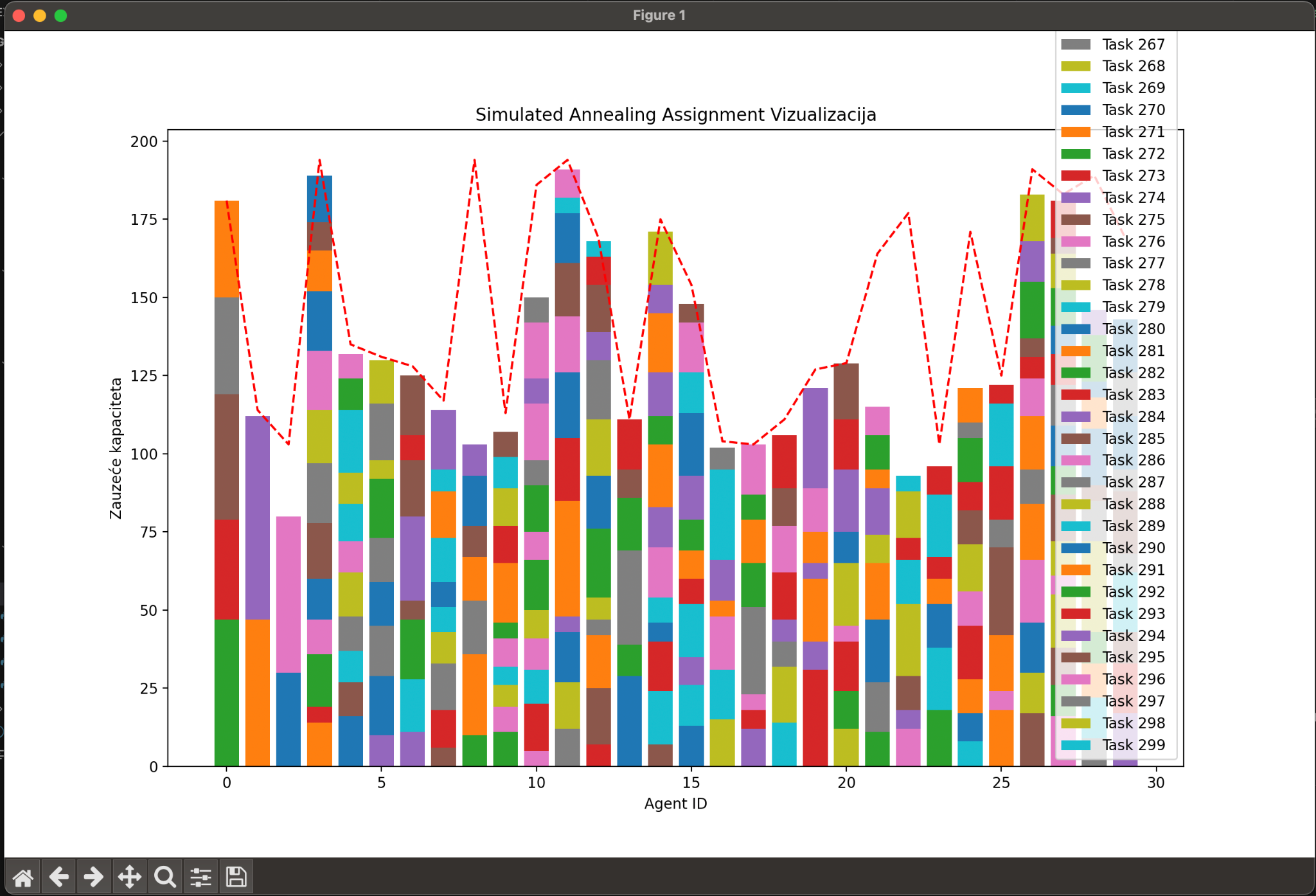
Za testiranje smo generisali instancu sa sledećim karakteristikama:

* **30 agenata**, kapaciteta između 100 i 200.
* **300 zadataka**, sa troškovima i zahtevima koji zavise od agenta.
* Prva tri agenta imaju **veoma niske troškove** ali **velike zahteve** → greedy algoritam će ih brzo popuniti, što može voditi u lošiju ukupnu dodelu.

### **3.3 Kvantitativni rezultati**

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

### **3.4 Grafička poređenja**



### **3.5 Diskusija**

Rezultati potvrđuju da:

* **random rešenje** služi kao referentna tačka i pokazuje koliko loše rešenje može izgledati bez ikakve optimizacije.
* **greedy pristup** daje solidna rešenja vrlo brzo, ali se oslanja isključivo na lokalni izbor bez preispitivanja prethodnih odluka.
* **simulated annealing** ima potencijal da značajno poboljša rešenje, naročito na težim instancama — ali po cenu većeg vremena izvršavanja.

U slučajevima gde greedy napravi "rane greške", SA ih uspešno ispravlja kroz iterativnu pretragu i stohastičko prihvatanje rešenja.

## **4. Zaključak**

U ovom radu analizirali smo tri pristupa rešavanju Generalizovanog problema dodele zadataka (GAP): nasumičnu dodelu (random assignment), heurističku greedy metodu i metaheuristički algoritam simuliranog kaljenja (Simulated Annealing – SA).

Nasumična dodela je poslužila kao donja referentna tačka, pokazavši kako izgleda rešenje bez ikakve optimizacije. Greedy pristup je dao kvalitetnija rešenja u znatno kraćem vremenu, ali se pokazao ograničenim u instancama koje zahtevaju globalno planiranje dodele. Simulated Annealing, iako vremenski zahtevniji, uspeo je da pronađe kvalitetnija rešenja u poređenju sa oba prethodna pristupa — naročito na velikim i "namerno nepovoljnim" instancama za greedy heuristiku.

Ključna prednost SA algoritma ogleda se u njegovoj sposobnosti da napušta lokalne minimume zahvaljujući stohastičkom prihvatanju lošijih rešenja u ranoj fazi pretrage. Time je u stanju da ispravi greške greedy pristupa i da pronađe rešenja koja značajno smanjuju ukupni trošak.

Sa druge strane, SA zahteva fino podešavanje hiperparametara (temperatura, broj iteracija, način generisanja susednih rešenja) i duže vreme izvršavanja, što može biti nepraktično za real-time sisteme ili jako velike instance bez dodatne optimizacije.

### **Pravci daljeg rada**

Za buduće proširenje, moguće je razmotriti sledeće pravce:

* Implementacija **Tabu Search** algoritma, koji sistematski izbegava vraćanje u prethodno posećena rešenja.
* Korišćenje **Genetskog algoritma** sa ukrštanjem i mutacijom nad populacijom rešenja.
* **Paralelizacija SA algoritma**, kako bi se višestruke nezavisne pretrage pokretale simultano i razmenjivale informacije.
* Testiranje na **realnim GAP instancama iz literature**, kao što su one iz OR-Library.

## **5. Literatura**

1. Chu, P. C., & Beasley, J. E. (1997). *A genetic algorithm for the generalized assignment problem*. Computers & Operations Research, 24(1), 17–23. https://doi.org/10.1016/S0305-0548(96)00024-3
2. 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. https://doi.org/10.1016/j.ejor.2004.09.059
3. Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). *Optimization by simulated annealing*. Science, 220(4598), 671–680. https://doi.org/10.1126/science.220.4598.671
4. Osman, I. H., & Laporte, G. (1994). *Metaheuristics: A bibliography*. Annals of Operations Research, 63(5), 513–628. https://doi.org/10.1007/BF02125436
5. Burkard, R. E., Çela, E., Pardalos, P. M., & Pitsoulis, L. S. (1999). *The Quadratic Assignment Problem*. In Handbook of Combinatorial Optimization (pp. 241–338). Springer. https://doi.org/10.1007/978-1-4613-0303-9\_5
6. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed.). MIT Press. [Koristi se za formalnu definiciju problema i teorijsku osnovu heuristika.]