Izrada programske aplikacije za rješavanje problema raspoređivanja pomoću mađarske metode

Lesar, Hrvoje

Undergraduate thesis / Završni rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:548060

Rights / Prava: Attribution 3.0 Unported/Imenovanje 3.0

Download date / Datum preuzimanja: **2024-09-21**

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Repository / Repozitorij:

Faculty of Organization and Informatics - Digital Repository

zir.nsk.hr

DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

VARAŽDIN

Hrvoje Lesar

IZRADA PROGRAMSKE APLIKACIJE ZA RJEŠAVANJE PROBLEMA RASPOREĐIVANJA POMOĆU MAĐARSKE METODE

ZAVRŠNI RAD

Varaždin, 2021.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Hrvoje Lesar

Matični broj: 46050/17-R

Studij: Poslovni sustavi

IZRADA PROGRAMSKE APLIKACIJE ZA RJEŠAVANJE PROBLEMA RASPOREĐIVANJA POMOĆU MAĐARSKE METODE

ZAVRŠNI RAD

Mentorica :

Doc. dr. sc. Nikolina Žajdela Hrustek

Varaždin, lipanj 2021.

Hrvoje Lesar

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Mađarska metoda je jedna od metoda za rješavanja problema raspoređivanja koji je posebni slučaj transportnog problema. Kroz rad je opisano linearno programiranje, transportni problem, problem raspoređivanja, mađarska metoda, te prikazan primjer korištenja mađarske metode. Rad također sadrži prikaz izrađene programske aplikacije koja služi za rješavanje problema raspoređivanja i koristi mađarsku metodu za postizanje rezultata.

Ključne riječi : problem raspoređivanja; mađarska metoda; problem asignacije; programska aplikacija;

Sadržaj

1. Uvod

Tema završnog rada je izrada programske aplikacije za rješavanje problema raspoređivanja pomoću mađarske metode iz čega proizlazi i osnovni cilj rada, a to je izrađena programska aplikacija koja omogućuje korisniku na jednostavan način rješavanje problema raspoređivanja. Ovaj rad sastoji se od dva dijela, teorijski dio, u kojem je dana teorijska osnova vezana uz metodu koja je poslužila kao osnova za izradu praktičnog dijela koji sadrži arhitekturu i praktičnog dijela prikaza izrade i rada osmišljene aplikacije.

Teorijski dio rada započinje osnovama linearnog programiranja, odnosno koracima postavljanja problema linearnog programiranja. U narednim poglavljima opisan je transportni problem, prikazane vrste i modeli transportnog problema te problem raspoređivanja koji se izvodi iz transportnog problema. Zatim slijedi definiranje samog problema raspoređivanja, koraci izrade modela te su ukratko prikazane i navedene osim mađarske metode i druge metode za rješavanje problema raspoređivanja. U zadnjem poglavlju teorijskog dijela detaljnije je opisana mađarska metoda koja je poslužila kao osnova za izradu programske aplikacije. U tom poglavlju su dani koraci za korištenje mađarske metode, te je obrađen jedan primjer koji dosljedno prikazuje što se događa u svakom koraku metode i kako se postupno matrica koeficijenata mijenja sve do dolaženja do rješenja.

Praktični dio rada fokusiran je na opis koraka izrade programske aplikacije. Sam opis sadrži detaljni prikaz arhitekture aplikacije, aktivnosti koje se odvijaju u aplikaciji tokom rada te alate i tehnologije korištene za izradu programske aplikacije. Za potrebe aplikacije razvijena je knjižnica za programski jezik Rust koja sadrži sve potrebne strukture i funkcije za rješavanje problema raspoređivanja pomoću mađarske metode. Kako bi aplikacija bila što lakša za upotrebu od strane korisnika izrađena je u tipu web aplikacije s grafičkim sučeljem. Razlog tome je što su web aplikacije uglavnom više pristupačne korisnicima nego klasične aplikacije koje je potrebno instalirati na računalo. U preostalim poglavljima rada dan je prikaz same aplikacije, prikazani su i opisani svi ključni elementi sučelja, kreirane su kratke upute i opisi korisni za krajnjeg korisnika kojima se pojašnjava korištenje različitih dostupnih opcija aplikacije. Između ostalog prikazani su određeni problemi i izazovi u razvoju aplikacije, te je na samom kraju dan i primjer korištenja aplikacije i dobivanja konačnog rješenja.

2. Linearno programiranje

Linearno programiranje je jedan od matematičkih alata za rješavanje problema optimizacije i vrlo je često korišten u velikim industrijama, kao što su bankarstvo, naftna industrija, šumarstvo i obrazovanje, za optimiziranje poslovanja [1]. Barković [2] definira linearno programiranje kao "rješavanje nekog matematičkog zadatka koji se sastoji u optimiranju neke linearne funkcije čije varijable zadovoljavaju neki linearni sistem jednadžbi i nejednadžbi". Iz toga možemo zaključiti da je linearno programiranje rješavanje nekog problema u cilju dobivanja optimalnog rješenja, a teži se prema postizanju minimalne (npr. troškovi proizvodnje) ili maksimalne (npr. dobit kod prodaje) vrijednosti, korištenjem sustava linearnih jednadžbi.

Prema Winston i Goldberg-u karakteristike koje dijele svi problemi linearnog programiranja i kojima možemo definirati problem linearnog programiranja su [1]:

1. Funkcija cilja, funkcija koja se sastoji od varijabli odluke i koju pokušavamo maksimizirati ili minimizirati,
2. Ograničenja, kojima varijable odluke moraju odgovarati, te ograničenja moraju biti linearna jednadžba ili nejednadžba,
3. Ograničenja znakova jednakosti i nejednakosti, kojima se svakoj varijabli određuje moguće vrijednosti varijable.

Prema navedenim karakteristikama možemo problem linearnog programiranja izraziti u općem obliku kao [3]:

$$\text{minimizirati } f(x) \tag{2.1}$$

$$\text{uz ograničenja } g_i(x) \geq 0 \; i = 1,\ldots,m \tag{2.2}$$

$$h_j(x) = 0 \; j = 1,\ldots,p \tag{2.3}$$

gdje su f, g_i, h_j generalne funkcije od parametra $f \in \mathrm{IR}$. U ovom obliku prepoznajemo sve navedene karakteristike: $f(x)$ kao funkciju cilja koju minimiziramo, $g_i(x)$ i $h_j(x)$ kao ograničenja i znakove = i ≥ kao ograničenja znakova jednakosti i nejednakosti. Tehnike za rješavanje linearnog problema su skoro uvijek iterativne te time možemo spomenuti jednu od najkorištenijih metoda za rješavanje problema linearnog programiranja, simpleks algoritam.

2.1. Transportni problem

Transportni problem je posebni tip problema linearnog programiranja kod kojeg je cilj minimalizirati cijenu distribucije proizvoda od ishodišta do određenog broja odredišta [4]. Ovakav problem je specificiran sljedećim pretpostavkama [1]:

1. Skupom od *m* ishodišta iz kojih se proizvod šalje, te ishodište *i* može ponuditi najviše s_i jedinica proizvoda.
2. Skupom od *n* odredišta prema kojima proizvod putuje i odredište *j* mora primiti barem d_j jedinica poslanog proizvoda.
3. Svaka jedinica proizvoda proizvedena na ishodištu *i* te poslana na odredište *j* stvara varijabilnu cijenu c_{ij}.

Opći model zatvorenog transportnog problema [2, str 117-118]:

$$Z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

Ograničenja za ishodišta

$$\sum_{j=1}^{n} x_{ij} = a_i \, (i = 1, 2, \dots, m)$$

Ograničenja za odredišta

$$\sum_{i=1}^{m} x_{ij} = b_j \, (j = 1, 2, \dots, n)$$

Uvjet nenegativnosti:

$$x_{ij} \geq 0 \begin{cases} za\ i = 1, 2, \dots, m \\ za\ j = 1, 2, \dots, n \end{cases}$$

Transportni problem je zatvoreni ako vrijedni da je suma količina u ishodištu jednaka sumi količina u odredištu:

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$$

Korištenjem prije navedenih triju pretpostavki i općeg modela transportnog problema ćemo kroz slijedeći primjer prikazati formuliranje modela za transportni problem.

Tablica 1: Ponuda, potražnja i jedinična cijena transporta (Autorski rad)

	Odredište				
Ishodište	Skladište 1	Skladište 2	Skladište 3	Skladište 4	Ponuda
Pogon 1	50	24	48	33	15
Pogon 2	10	38	49	42	38
Pogon 3	17	29	14	25	27
Potražnja	36	18	10	16	Ukupno: 80

Izvor: Izrada autora rada

U tablici 1 su prikazani pogoni, skladišta, cijena transporta između određenog pogona i skladišta te ponuda pogona i potražnja skladišta. Tako iz tablice možemo prepoznati skup ishodišta m = {*Pogon* 1*, Pogon* 2*, Pogon* 3}, te njihovu najvišu ponudu jedinica proizvoda $s_1 = 15$, $s_2 = 38$, $s_3 = 27$. Isto prepoznajemo skup odredišta n = {*Skladište* 1*, Skladište* 2*, Skladište* 3*, Skladište* 4} i potražnju odredišta $d_1 = 36$, $d_2 = 18$, $d_3 = 10$, $d_4 = 16$ i na kraju vidimo varijabilnu cijenu transporta od ishodišta do odredišta (npr. $c_{11} = 50$, $c_{31} = 17$). Zadani problem nije potrebno zatvarati pošto je zbroj ponuda po pogonima i zbroj potražnje po skladištima jednak.

Ukoliko zbrojevi nisu jednaki razlikujemo dvije vrste problema [4]:

1. Otvoreni transportni problem s viškom u ponudi,
2. Otvoreni transportni problem s viškom u potražnji.

Takve specifične vrste problema potrebno je prije rješavanja pretvoriti u zatvoreni i uravnotežiti na način da se uvodi fiktivno ishodište ili odredište koje isporučuje manjak tj. preuzima višak u slučaju odredišta [2]. Kako je ponuda i potražnja u primjeru uravnotežena možemo odrediti funkciju cilja prema cijeni transporta zadanog u tablici 1:

$$50x_{11} + 24x_{12} + 48x_{13} + 33x_{14} \text{ (Cijena transporta od Pogona 1)}$$

$$+10x_{21} + 38x_{22} + 49x_{23} + 42x_{24} \text{ (Cijena transporta od Pogona 2)}$$

$$+17x_{31} + 29x_{32} + 14x_{33} + 25x_{34} \text{ (Cijena transporta od Pogona 3)}$$

Sada možemo odrediti ograničenja za svako ishodište, gdje x_{ij} predstavlja dostavu od pogona *i* prema skladištu *j*:

$$x_{11} + x_{12} + x_{13} + x_{14} = 15 \text{ (Ograničenje ponude za Pogon 1)}$$

$$x_{21} + x_{22} + x_{23} + x_{24} = 38 \text{ (Ograničenje ponude za Pogon 2)}$$

$$x_{31} + x_{32} + x_{33} + x_{34} = 27 \text{ (Ograničenje ponude za Pogon 3)}$$

Slijedeće moramo postaviti ograničenja za skladišta tako da svako skladište primi dovoljno proizvoda za pokrivanje potražnje:

$$x_{11} + x_{21} + x_{31} = 36 \text{ (Ograničenje potražnje za Skladište 1)}$$

$$x_{12} + x_{22} + x_{32} = 18 \text{ (Ograničenje potražnje za Skladište 2)}$$

$$x_{13} + x_{23} + x_{33} = 10 \text{ (Ograničenje potražnje za Skladište 3)}$$

$$x_{14} + x_{24} + x_{34} = 16 \text{ (Ograničenje potražnje za Skladište 4)}$$

Zbog toga što sve vrijednosti x_{ij} moraju biti pozitivne dodajemo još i uvjet nenegativnosti $x_{ij} \geq 0$. Na posljetku kombiniramo funkciju cilja, ograničenja ishodišta, ograničenja odredišta i uvjet nenegativnosti kako bi dobili model problema linearnog programiranja za transportni problem kod zadanog primjera:

$$\min z = 50x_{11} + 24x_{12} + 48x_{13} + 33x_{14} + 10x_{21} + 38x_{22} + 49x_{23} + 42x_{24} + 17x_{31} + 29x_{32} + 14x_{33} + 25x_{34}$$

(Ograničenja ponude)

$$x_{11} + x_{12} + x_{13} + x_{14} = 15$$

$$x_{21} + x_{22} + x_{23} + x_{24} = 38$$

$$x_{31} + x_{32} + x_{33} + x_{34} = 27$$

(Ograničenja potražnje)

$$x_{11} + x_{21} + x_{31} = 36$$

$$x_{12} + x_{22} + x_{32} = 18$$

$$x_{13} + x_{23} + x_{33} = 10$$

$$x_{14} + x_{24} + x_{34} = 16$$

$$x_{ij} \geq 0$$

Postavljeni model se koristi za rješavanje problema koristeći simpleks metodom ili metodama koje su više efikasne za rješavanje transportnih problema kao što je modificirana metoda distribucije (MODI metoda). MODI metoda je zapravo simpleks metoda koja je modificirana i prilagođena za rješavanje transportnih problema [4].

2.2. Problem raspoređivanja

Problem raspoređivanja je posebni slučaj transportnog problema kod kojeg su vrijednosti koje se nude i traže jednake jedan [2]. Ovakav problem je karakteriziran znanjem o cijeni dodjeljivanja *n* poslova na *n* mjesta uz uvjet da točno jedan posao može biti dodijeljen na točno jedno mjesto. Cilj je postizanje optimalnog rasporeda poslova i mjesta kako bi se postigao maksimalni učinak uz minimalne troškove.

2.2.1. Izgradnja modela

Kako bi izgradili opći model problema raspoređivanja uvodi se binarna matrica $X = (x_{ij})$ tako da vrijedi [6, str. 5-6]:

$$x_{ij} = \begin{cases} 1, ako\ je\ red\ i\ dodijeljen\ stupcu\ j, \\ 0, u\ suprotnom\ slučaju \end{cases}$$

i sad modeliramo problem kao:

$$\min z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad (i = 1, 2, \dots, n)$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad (j = 1, 2, \dots, n)$$

$$x_{ij}\ \epsilon\ 0, 1\ (i, j = 1, 2, \dots, n)$$

time kao rezultat dobivamo matricu *X* koja je matrica permutacija.

2.2.2. Algoritmi za rješavanje

Pošto je problem raspoređivanja zapravo samo posebni tip transportnog problema moguće ga je rješavati metodama i algoritmima koji su pogodni za rješavanje transportnog problema, te općenito algoritmima za rješavanje problema linearnog programiranja (simpleks). Zbog specifičnosti problema raspoređivanja i visokog stupnja degeneracije algoritmi za

rješavanje transportnih problema nisu toliko efikasni ni pogodni za rješavanje, te su time razvijeni posebni algoritmi koji se fokusiraju na efikasno rješavanje ovakvog problema.

Veliki je broj metoda i algoritama pogodnih za rješavanje ovakvog problema, određeni algoritmi samo nadograđuju druge kako bi bili što efikasniji, te ćemo ovdje nabrojiti i ukratko opisati neke od tih algoritama i metoda. Možda najpoznatija i najkorištenija metoda je mađarska metoda koja se smatra kao prethodnik primal - dual metode [6]. Kao nadogradnja na mađarsku metodu implementiran je algoritam najkraćeg puta za mađarsku metodu kojim se postignula bolja efikasnost metode. Dinic–Kronrod algoritam vrlo efikasno rješava problem raspoređivanja te se postavlja kao dualni problem [7]. 1980. godine Ming S. Hung i Walter O. Rom objavljuju njihov novi algoritam za rješavanje problema raspoređivanja koji se bazira na shemi relaksiranja zadanog problema u niz jednostavnih mrežnih (transportnih) problema za koje je jednostavno pronaći rješenje [8]. M. L. Balinski nadograđuje metodu Hunga i Roma korištenjem duala simpleks metode.

2.2.3. Usporedba algoritama

U ovom poglavlju ćemo ukratko kategorizirati i navesti vremensku složenost nekih od algoritama iz poglavlja 2.2.2. Algoritmi će biti kategorizirani po bazičnom pristupu rješavanja problema, a vremenska složenost će biti prikazana korištenjem O notacije.

Tablica 2: Usporedba algoritama

Naziv/Autor	Vremenska složenost	Kategorija	Komentar
Mađarska metoda -original	$O(n^4)$	Primal-dual	
Mađarska metoda – najkraći put	$O(n^3)$	Najkraći put	Modifikacija originalne mađarske metode za postizanje bolje efikasnosti
Dinic-Kronrod algoritam	$O(n^3)$	Dual	
Hung i Rom	$O(n^3)$	Dual	
M. L. Balinski	$O(n^3)$	Dual simpleks	
Cunningham	$O(c^n)$	Primalni simpleks	Velika vremenska složenost zbog visokog stupnja degeneracije
Gabow	$O(n^{3/4} m \log C)$	Skaliranje troškova	$C = \max_{i,j}\{c_{ij}\}$, skaliranje troškova

			bazirano na mađarskoj metodi
Kao, Lam, Sung i Ting	$O(\sqrt{n}W \log \frac{n^2}{\frac{W}{C}} / \log n)$	Dekompozicija	$W = P_{\|i,j\| \in E} c_{ij}$

Izvor: Burkard, Dell'Amico i Martello,Assignment Problems

Vremenska kompleksnost Cunningham-ovog algoritma je dobar primjer zašto sama simpleks metoda nije prigodna za rješavanje problema raspoređivanja, te vidimo kako ostale metode koje se fokusiraju samo na rješavanje jednog problema postižu bolje vremensku složenost.

3. Mađarska metoda

U daljnjim potpoglavljima se opisuje razvoj mađarske metode, dobiveni koraci za rješavanje problema raspoređivanja ovom metodom, te primjer korištenja. Primjer u detalje prikazuje korištenje mađarske metode za rješavanje problema raspoređivanja i prati zadane korake rješavanja.

3.1. Opis i razvoj metode

Godine 1953. Harold W. Kuhn je čitao knjigu o teoriji grafova mađarskog autora Dénes Kőnig-a gdje je prepoznao Kőnig-ov teorem kao primjer dualnosti, te teorem glasi: „Ako su brojevi u matrici nule i jedinice, tada maksimalni broj redova i stupaca koji će sadržati sve jedinice je jednak maksimalnom broju jedinica koje mogu biti odabrane na način da nisu odabrane dvije u istom redu ili stupcu“ [9, str 642]. U istoj knjizi Kuhn pronalazi napomenu u kojoj Kőnig upućuje na rad E. Egerváry-a koji daje računski trivijalnu metodu za smanjivanje općeg problema raspoređivanja na 0-1 problem (Problem ranca). Spajanjem tih dviju ideja Kuhn kreira mađarsku metodu, nazvanu prema dvama mađarskim matematičarima D. Kőnig-u i E. Egerváry-u [10].

3.2. Matematički model

Mađarska metoda za rješavanje problema raspoređivanja koristi zadanu matricu A = a_{ij} iz koje se izvede matrica B = b_{ij} prema [2]:

$$b_{ij} = a_{ij} - u_i - v_j \ (i = 1, \dots, n; j = 1, \dots, n)$$

gdje su u_i i v_j proizvoljno odabrane konstante tj. varijable dualnog problema. Prema tome mađarska metoda koristi dualnost problema raspoređivanja i s tim rješenje matrice A je jednako rješenju matrice B.

Problem raspoređivanja kod kojeg se traži minimalna vrijednost definiran u poglavlju 2.2.1. jednak je dualnom problemu [2]:

$$\max \bar{z} = \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j$$

uz uvjet $a_{ij} \geq u_i + v_j$

3.3. Koraci rješavanja problema

Problem raspoređivanja se ovom metodom može riješiti kroz tri koraka [4, str 57]:

1. Pronalazi se minimalni element u svakom redu *n* x *n* matrice troškova. Konstruira se nova matrica na način da se minimalni trošak reda oduzima od svakog elementa u tom redu. Za novu kreiranu matricu se ponavlja postupak samo ovaj put po svim stupcima umjesto redovima. Krajnja konstruirana matrica se naziva matrica reduciranih troškova, te kao takva u svakom redu i stupcu sadrži barem jednu nulu.
2. Potrebno je minimalnim brojem linija (horizontalnih, vertikalnih ili i jednih i drugih) precrtati sve nule u matrici reduciranih troškova. Ako je za to potrebno točno *n* linija tada moguće pronaći optimalno rješenje među precrtanim nulama, no ako je broj linija manji od *n* nastavlja se na treći korak.
3. Pronaći najmanji ne nulti element u matrici reduciranih troškova koji nije prekriven linijama nacrtanim u drugom koraku. Pronalaskom tog elementa, potrebno ga je oduzeti od svih linijama ne pokrivenih elemenata te ga dodati svim elementima pokrivenim s dvije linije, nakon toga se vratiti na drugi korak.

Neke napomene za korištenje navedenih koraka za rješavanje [1, str. 395]:

- Prvi korak zahtijeva da je matrica kvadratna, te kako bi se mogao riješiti otvoreni problem raspoređivanja potrebno ga je zatvoriti na način da se u matricu dodaju fiktivni elementi vrijednosti nula, ako je potrebno dodaju se i fiktivni redovi ili stupci s vrijednostima elemenata jednakim nulama.
- Kako bi se riješio problem raspoređivanja kojemu je cilj maksimizirati funkciju cilja potrebno je pomnožiti matricu s *-1* i riješiti problem minimalizacijom.
- U drugom koraku nije definiran način na koji se pronalazi minimalni broj linija i zbog toga kod rješavanja većih problema može biti teško pronaći minimalan broj linija.

3.4. Primjena mađarske metode na primjeru

U malom proizvodnom pogonu potrebno je rasporediti zaposlenike na proizvodna mjesta gdje će proizvoditi najmanju količinu škarta. Za svakoga zaposlenika je zabilježena prosječna količina škarta koji se napravi na pojedinom proizvodnom mjestu i ti podaci su prikazani u sljedećoj tablici.

Tablica 3: Prosječna količina škarta po zaposleniku na određenom proizvodnom mjestu

	Zaposlenici						
Proizvodna mjesta	Zaposlenik 1	Zaposlenik 2	Zaposlenik 3	Zaposlenik 4	Zaposlenik 5	Zaposlenik 6	Zaposlenik 7
Proizvodno mjesto 1	21	10	13	25	16	16	5
Proizvodno mjesto 2	16	12	23	25	16	4	24
Proizvodno mjesto 3	14	13	10	23	22	24	28
Proizvodno mjesto 4	11	23	16	28	25	11	24
Proizvodno mjesto 5	16	9	23	20	13	29	20
Proizvodno mjesto 6	4	17	9	14	11	12	24

Izvor: Izrada autora rada

Iz zadane tablice kreiramo matricu troškova na način da iz tablice ispišemo za svaki red pojedinu količinu škarta i time dobivamo sljedeću matricu:

$$\begin{bmatrix} 21 & 10 & 13 & 25 & 16 & 16 & 5 \\ 16 & 12 & 23 & 25 & 16 & 4 & 24 \\ 14 & 13 & 10 & 23 & 22 & 24 & 28 \\ 11 & 23 & 16 & 28 & 25 & 11 & 24 \\ 16 & 9 & 23 & 20 & 13 & 29 & 20 \\ 4 & 17 & 9 & 14 & 11 & 12 & 24 \end{bmatrix}$$

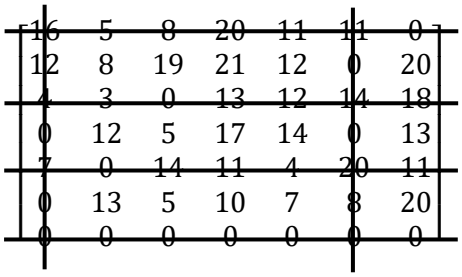
Možemo primijetiti da konstruirana matrica nije kvadratna, tj. zadani problem je otvoreni i potrebno ga je zatvoriti kako bi mogli primijeniti mađarsku metodu. Problem zatvaramo na način da dodajemo fiktivni red ili stupac u kojemu su sve vrijednosti elemenata jednake nuli, u ovom slučaju dodajemo fiktivni red koji kao rezultat daje kvadratnu matricu reda *7 x 7.*

$$\begin{bmatrix} 21 & 10 & 13 & 25 & 16 & 16 & 5 \\ 16 & 12 & 23 & 25 & 16 & 4 & 24 \\ 14 & 13 & 10 & 23 & 22 & 24 & 28 \\ 11 & 23 & 16 & 28 & 25 & 11 & 24 \\ 16 & 9 & 23 & 20 & 13 & 29 & 20 \\ 4 & 17 & 9 & 14 & 11 & 12 & 24 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Započinjemo metodu pronalaženjem minimalne vrijednosti u svakom redu i oduzimanja te vrijednosti od svakog elementa u redu. Tako u svakom redu dobivamo barem jednu nulu. Sljedeća matrica prikazuje rezultat oduzimanja minimalnog elementa od svih elemenata u redu i poseban prikaz vrijednosti za koju je svaki red bio oduzet:

$$\begin{bmatrix} 16 & 5 & 8 & 20 & 11 & 11 & 0 \\ 12 & 8 & 19 & 21 & 12 & 0 & 20 \\ 4 & 3 & 0 & 13 & 12 & 14 & 18 \\ 0 & 12 & 5 & 17 & 14 & 0 & 13 \\ 7 & 0 & 14 & 11 & 4 & 20 & 11 \\ 0 & 13 & 5 & 10 & 7 & 8 & 20 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} 5 \\ 4 \\ 10 \\ 11 \\ 9 \\ 4 \\ 0 \end{matrix}$$

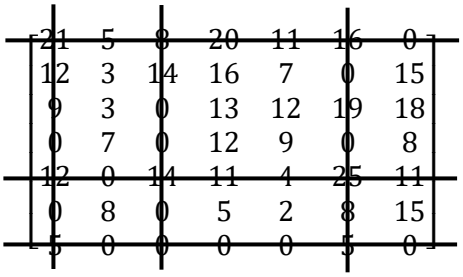
Isto tako činimo i za svaki stupac u matrici, ali pošto imamo fiktivni red minimalna vrijednost u svakom stupcu će biti nula i matrica se neće promijeniti, time smo dobili matricu reduciranih troškova. Nastavljamo na sljedeći korak kod kojeg moramo odrediti minimalni broj linija kojima možemo pokriti sve nule u matrici.



Ukupni broj linija iznosi *6* što je manje od broja potrebnih linija *7* te nastavljamo na sljedeći korak. Pronalazimo najmanji element u matrici koji nije prekriven linijom, u ovom slučaju je to element vrijednosti *5*. Pronađeni element oduzimamo od svih elemenata koji također nisu pokriveni linijama i dodajemo svim elementima koji su pokriveni s dvije linije. Sljedeća matrica je rezultat oduzimanja i dodavanja pronađenog elementa.

$$\begin{bmatrix} 21 & 5 & 8 & 20 & 11 & 16 & 0 \\ 12 & 3 & 14 & 16 & 7 & 0 & 15 \\ 9 & 3 & 0 & 13 & 12 & 19 & 18 \\ 0 & 7 & 0 & 12 & 9 & 0 & 8 \\ 12 & 0 & 14 & 11 & 4 & 25 & 11 \\ 0 & 8 & 0 & 5 & 2 & 8 & 15 \\ 5 & 0 & 0 & 0 & 0 & 5 & 0 \end{bmatrix}$$

Na dobivenoj matrici ponavljamo drugi korak i minimalnim brojem linija pokrivamo sve nule.



Opet vidimo da broj linija nije jednak *7* i ponavljamo treći korak. Minimalna vrijednost od ne pokrivenih elemenata je *2* i opet oduzimamo tu vrijednost od ne pokrivenih elemenata i dodajemo ju elementima pokrivenih sa dvije linije.

$$\begin{bmatrix} 23 & 5 & 10 & 20 & 11 & 18 & 0 \\ 12 & 1 & 14 & 14 & 5 & 0 & 13 \\ 9 & 1 & 0 & 11 & 10 & 19 & 16 \\ 0 & 5 & 0 & 10 & 7 & 0 & 6 \\ 14 & 0 & 16 & 11 & 4 & 27 & 11 \\ 0 & 6 & 0 & 3 & 0 & 8 & 13 \\ 7 & 0 & 2 & 0 & 0 & 7 & 0 \end{bmatrix}$$

Ponovo prelazimo na drugi korak.

$$\begin{bmatrix} 23 & 5 & 10 & 20 & 11 & 18 & 0 \\ 12 & 1 & 14 & 14 & 5 & 0 & 13 \\ 9 & 1 & 0 & 11 & 10 & 19 & 16 \\ 0 & 5 & 0 & 10 & 7 & 0 & 6 \\ 14 & 0 & 16 & 11 & 4 & 27 & 11 \\ 0 & 6 & 0 & 3 & 0 & 8 & 13 \\ 7 & 0 & 2 & 0 & 0 & 7 & 0 \end{bmatrix}$$

Sad je broj linija jednak *7* i sad smo sigurni da možemo iz pokrivenih nula pronaći optimalno rješenje. To radimo na način da prvo odaberemo nule koje nemaju ni jednu drugu nulu u istom redu ili stupcu, ako postoji više redova i stupaca koji imaju jednak broj nula proizvoljno odabiremo jednu nulu. Odabirom nule u redu ili stupcu prekrižimo sve ostale nule u istom redu i stupcu kako ne bi njih odabrali. Sljedeća matrica prikazuje odabrane nule.

$$\begin{bmatrix} 23 & 5 & 10 & 20 & 11 & 18 & [\mathbf{0}] \\ 12 & 1 & 14 & 14 & 5 & [\mathbf{0}] & 13 \\ 9 & 1 & [\mathbf{0}] & 11 & 10 & 19 & 16 \\ [\mathbf{0}] & 5 & 0 & 10 & 7 & 0 & 6 \\ 14 & [\mathbf{0}] & 16 & 11 & 4 & 27 & 11 \\ 0 & 6 & 0 & 3 & [\mathbf{0}] & 8 & 13 \\ 7 & 0 & 2 & [\mathbf{0}] & 0 & 7 & 0 \end{bmatrix}$$

Svaka odabrana nula označava vrijednost koju je potrebno odabrati u početnoj matrici (matrici troškova) kako bi se dobio minimalni trošak problema. U ovom slučaju odabrane nule označavaju sljedeće elemente u početnoj matrici:

$$\begin{bmatrix} 21 & 10 & 13 & 25 & 16 & 16 & [\mathbf{5}] \\ 16 & 12 & 23 & 25 & 16 & [\mathbf{4}] & 24 \\ 14 & 13 & [\mathbf{10}] & 23 & 22 & 24 & 28 \\ [\mathbf{11}] & 23 & 16 & 28 & 25 & 11 & 24 \\ 16 & [\mathbf{9}] & 23 & 20 & 13 & 29 & 20 \\ 4 & 17 & 9 & 14 & [\mathbf{11}] & 12 & 24 \\ 0 & 0 & 0 & [\mathbf{0}] & 0 & 0 & 0 \end{bmatrix}$$

Korištenjem odabranih vrijednosti možemo rasporediti zaposlenike na radna mjesta gdje će proizvesti minimalnu količinu škarta, ta minimalna količina iz matrice iznosi: *5 + 4 + 10 + 11 + 9 + 11 + 0 = 50*. Također možemo primijetiti da je jedan zaposlenik viška pošto imamo samo 6 proizvodnih mjesta, a 7 zaposlenika. Optimalni raspored radnika bi bio:

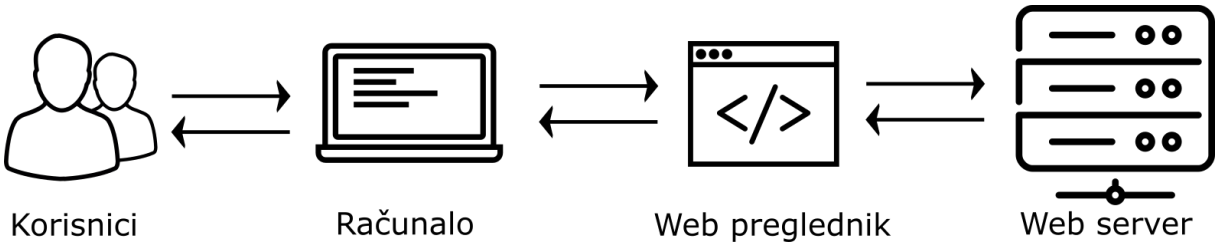
- Zaposlenik 1, proizvodno mjesto 4.
- Zaposlenik 2, proizvodno mjesto 5.
- Zaposlenik 3, proizvodno mjesto 3.
- Zaposlenik 4, proizvodno mjesto 7.
- Zaposlenik 5, proizvodno mjesto 6.
- Zaposlenik 6, proizvodno mjesto 2.
- Zaposlenik 7, proizvodno mjesto 1.

4. Izrada programske aplikacije

U ovom poglavlju rada je prikazana aplikacija koja rješava problem raspoređivanja mađarskom metodom. Za izradu dijela aplikacije koji je odgovoran za rješavanje problema je korišten programski jezik Rust. Razlog za korištenje jezika Rust je što nudi veliku brzinu rješavanja problema te time puno smanjuje vrijeme čekanja izračuna rezultata kod većih matrica. Aplikacija je podijeljena na dva dijela, prvi je web server koji korisniku daje sučelje za unos podataka, a drugi je sam program koji rješava zadani problem raspoređivanja i vraća krajnji rezultat. Oba dijela će biti opisana detaljnije u poglavljima koja slijede.

4.1. Arhitektura aplikacije

Kako je već prije navedeno aplikacija se sastoji od dva glavna dijela. Web servera i samog programa koji rješava zadani problem. Sljedeća slika prikazuje arhitekturu aplikacije. Arhitektura se sastoji od četiri dijela; korisnika, računala, web preglednika i web servera. Za korištenje aplikacije korisnik na računalu mora imati web preglednik kako bi mogao pristupiti aplikaciji. Kod prvog pristupanja aplikaciji web preglednik preuzima s web servera web stranicu i prikazuje ju korisniku. Potom korisnik ima mogućnost unosa matrice za problem raspoređivanja i slanja te unesene matrice na rješavanje na web server. Kod primitka problema web server rješava problem raspoređivanja korištenjem mađarske metode, te izračunom rezultata šalje dobiveni rezultat korisniku. Pri primitku rezultata web stranica korisniku prikazuje primljeni rezultat.



Slika 1: Arhitektura aplikacije

4.1.1. Zahtjevi i karakteristike aplikacije

Aplikacija je karakteristična po tome što se sastoji od dva dijela, web servera i knjižnice koja se sastoji od struktura i funkcija za rješavanje problema raspoređivanja. Razlog za razdjelu aplikacije je što web server korisniku pruža grafičko sučelje i po potrebi poziva funkcije iz knjižnice za rješavanje problema raspoređivanja kako bi korisniku prikazao rješenje. Prednost odvojenosti je što ako se nešto mijenja u knjižnici za rješavanje to neće utjecati na web server i obrnuto. Izrađena knjižnica je dostupna bilo kome tako da i drugi programeri mogu po potrebi koristiti knjižnicu kod razvoja aplikacija.

Funkcionalni zahtjevi:

- Unos matrica varijabilnih veličina – korisniku je bitno da može unositi matrice bilo kakvih veličina i dobiti rezultat.
- Minimalizacija, maksimalizacija rezultata – ovisno o tipu problema, mogućnost dobivanja rješenja za minimalne troškove ili maksimalni profit

Nefunkcionalni zahtjevi:

- Performanse – se odnose brzinu odziva aplikacije. U slučaju ove aplikacije najbitnije su performanse kod rješavanja problema, kako bi korisnik morao što manje vremena čekati za ispis rezultata.
- Korisničko sučelje – treba biti prilagođeno potrebama korisnika, intuitivno i jednostavno. Kod mnogih elemenata aplikacije postoje tipke za prikaz više informacija o elementu.
- Održivost – mjera po kojoj se očituje kako je lako mijenjati, održavati i poboljšavati aplikaciju.

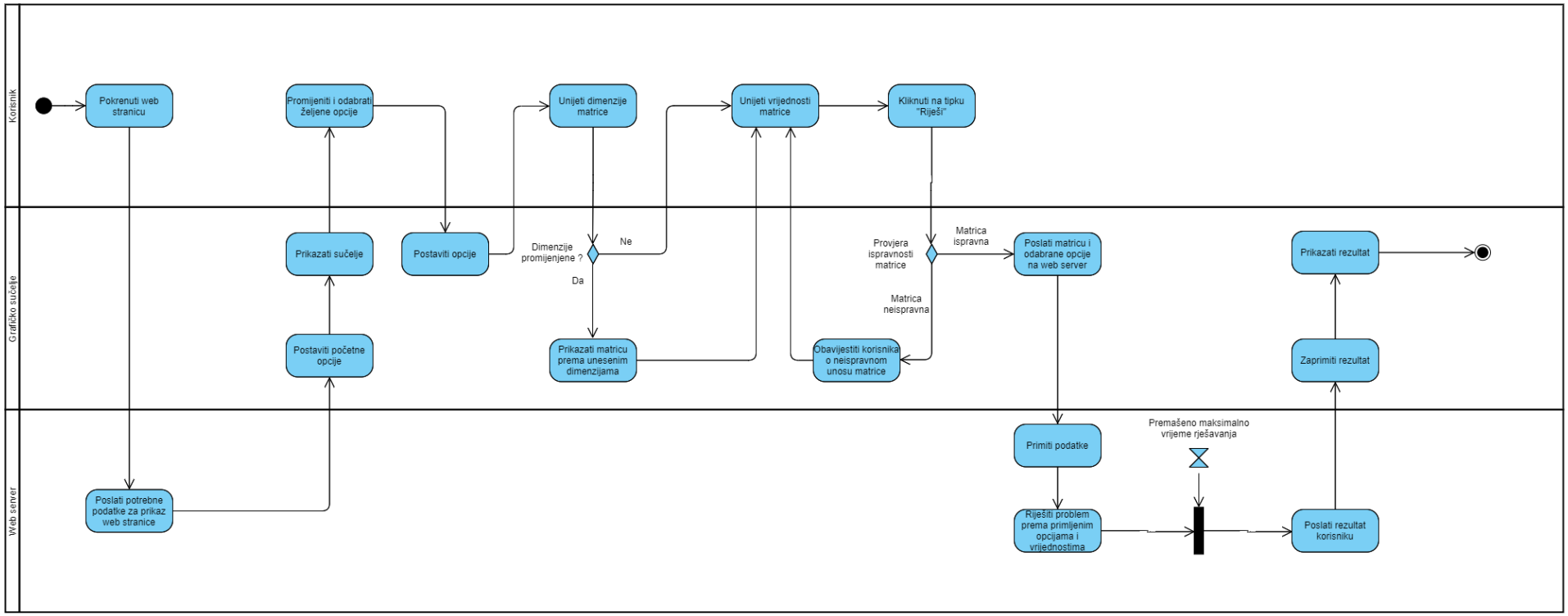
4.1.2. Ograničenja aplikacije

Za pristup aplikaciji je potreban pristup internetu. Zbog toga što je aplikacija dostupna na webu postavljena su neka ograničenja na web server. Ograničena je veličina matrice koja može biti poslana na web server kako bi se spriječilo zagušivanje servera. Maksimalno vrijeme rješavanja pojedinog problema je postavljeno na 10 sekundi što bi trebalo biti puno previše i za rješavanje većih matrica, no ako rješavanje problema potraje

duže od 10 sekundi web server prekida rješavanje i obavještava korisnika o prekoračenom maksimalnom vremenu rješavanja.

4.1.3. Dijagram aktivnosti aplikacije

Dijagram aktivnosti grafički prikazuje sam rad aplikacije i interakciju dijelova aplikacije s korisnikom. Kod otvaranja web stranice na kojoj je postavljena aplikacija korisnik od web servera zatraži prikaz grafičkog sučelja, te server odgovara s svim potrebnim podacima za prikaz sučelja i sučelje dolazi s već postavljenim opcijama koje korisnik kasnije može mijenjati. Korisnik može po želji postaviti dostupne opcije; način unosa matrice, metodu rješavanja problema raspoređivanja, funkciju cilja. Svaka od opcija je detaljnije opisana i objašnjena u poglavlju 4.2. Nakon postavljanja željenih opcija korisnik unosi dimenzije matrice, prema unesenim dimenzijama grafičko sučelje prikazuje određen broj ćelija. U ćelije se unose vrijednosti matrice, kad korisnik završi unos vrijednosti matrice može poslati problem na rješavanje na web server klikom na tipku „Riješi“. Klikom na tipku „Riješi“ unesena matrica i postavljene opcije su poslane na web server gdje se prema odabranim opcijama rješava problem. Kad web server izračuna rezultat ili se premaši maksimalno dostupno vrijeme rješavanja, ovisno koji slučaj se dogodi prije, web server šalje rezultat korisniku. Grafičko sučelje prikazuje primljeni rezultat kojeg korisnik može analizirati.



Slika 2: Dijagram aktivnosti aplikacije

4.1.4. Korišteni alati u izradi aplikacije

Alati korišteni za izradu aplikacije su Visual Studio Code, Git, Rust standardna knjižnica, Actix Web framework, Visual Paradigm. Visual Studio Code je korišten za pisanje i uređivanje koda aplikacije Git za pohranu izvornog koda i verzioniranje aplikacije. Standardna knjižnica programskog jezika Rust za osnovne strukture i funkcije potrebne za izradu programa. Actix Web služi kao web server koji komunicira s korisnikom i poslužuje grafičko sučelje korisniku te zaprima zahtjeve korisnika. Visual Paradigm je korišten za izradu dijagrama.

4.2. Prikaz programske aplikacije

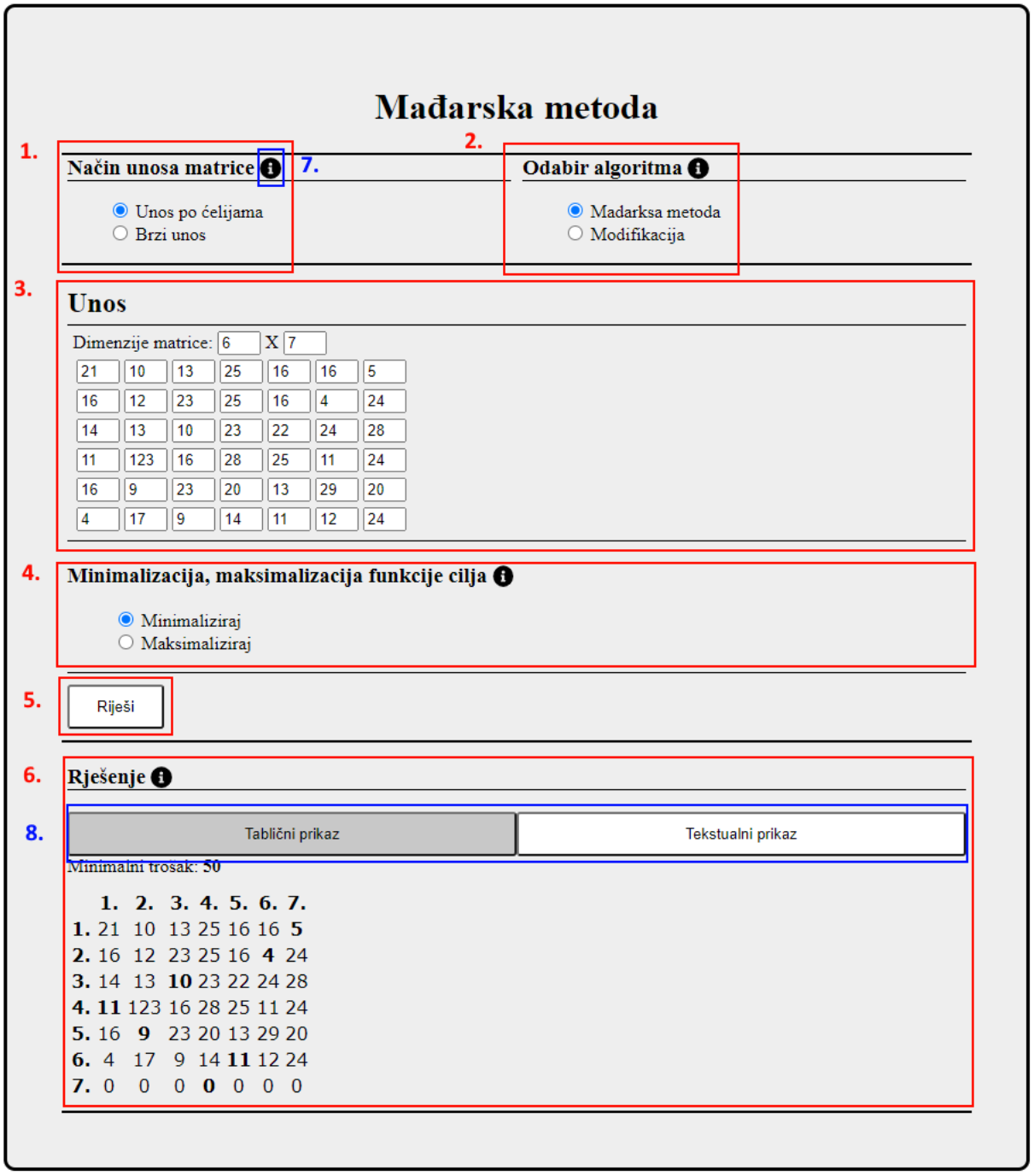
Kod pokretanja aplikacije korisniku je vidljivo sučelje prikazano na slici 3. Sučelje sadrži više opcija koje korisnik može dobrovoljno odabrati i promijeniti, dok su određene opcije postavljene već kod pokretanja aplikacije.

Slika 3: Početni izgled aplikacije

4.2.1. Elementi aplikacije

Na slijedećoj slici (Slika *4*.) vidljivi su elementi od kojih se aplikacije sadrži, svi elementi će biti detaljnije opisani u sljedećim poglavljima. Prikazani elementi su:

1. Način unosa matrice – korisnik bira između dva ponuđena načina unosa, unos po ćelijama i brzi unos.
2. Odabir algoritma – korisnik ima na odabir dva algoritma, originalnu mađarsku metodu i Munkres-ovu modifikaciju mađarske metode.
3. Unos – ovisno o odabranom načinu unosa korisnik unosi željenu matricu.
4. Minimalizacija, maksimalizacija funkcije cilja – određuje ako se funkcija cilja minimalizira ili maksimalizira.
5. Tipka „Riješi“ – rješava unesenu matricu prema odabranom algoritmu i funkciji cilja.
6. Rješenje – prikazuje korisniku minimalni trošak odnosno maksimalni profit riješenog problema, te nudi dvije vrste prikaza rezultata.
7. Tipka za više informacija – kroz aplikaciju je postavljeno više tipki koje korisnik može kliknuti i otvaraju novi prozor u kojem je detaljnije opisani npr. odabir algoritma.
8. Tipke za odabir načina prikaza rješenja – korisnik može mijenjati način prikaza rješenja iz tabličnog prikaza u tekstualni prikaz i obrnuto.



Slika 4: Elementi aplikacije

4.2.2. Način unosa matrice

Aplikacija nudi korisniku odabir dvije vrste unosa matrice:

- Unos po ćelijama - pri prvom odabiru prikazuje matricu dimenzije 5x5 no korisnik može unijeti bilo koje dimenzije matrice. Kod promjene dimenzija matrice dinamično se mijenja broj dostupnih ćelija koje predstavljaju po jedno polje u matrici. Ova funkcionalnost je predviđena da se koristi za unos manjih matrica pošto je potrebno ručno upisati broj u svaku ćeliju.

- Brzi unos - kod odabira brzog unosa sučelje se mijenja i korisnik dobiva prazno mjesto za unos teksta Korisnik može unijeti bilo kakav tekst no program prima samo validne matrice. Primjer izgleda brzog unosa je prikazan na slici 5. Brzi unos je namijenjen za unos većih matrica koje su već formatirane na način da svaki red sastoji od *n* brojeva odvojenih razmakom.

Korisnik ima mogućnost unosa bilo kakve matrice pa tako unesena matrica ne mora biti kvadratna. Kako bi se problem mogao riješiti mađarskom metodom matrica mora biti kvadratna, no aplikacija ne traži korisnika na unos kvadratne matrice već automatski pokuša pretvoriti unesenu matricu u kvadratnu dodavanjem fiktivnih redova ili stupaca.

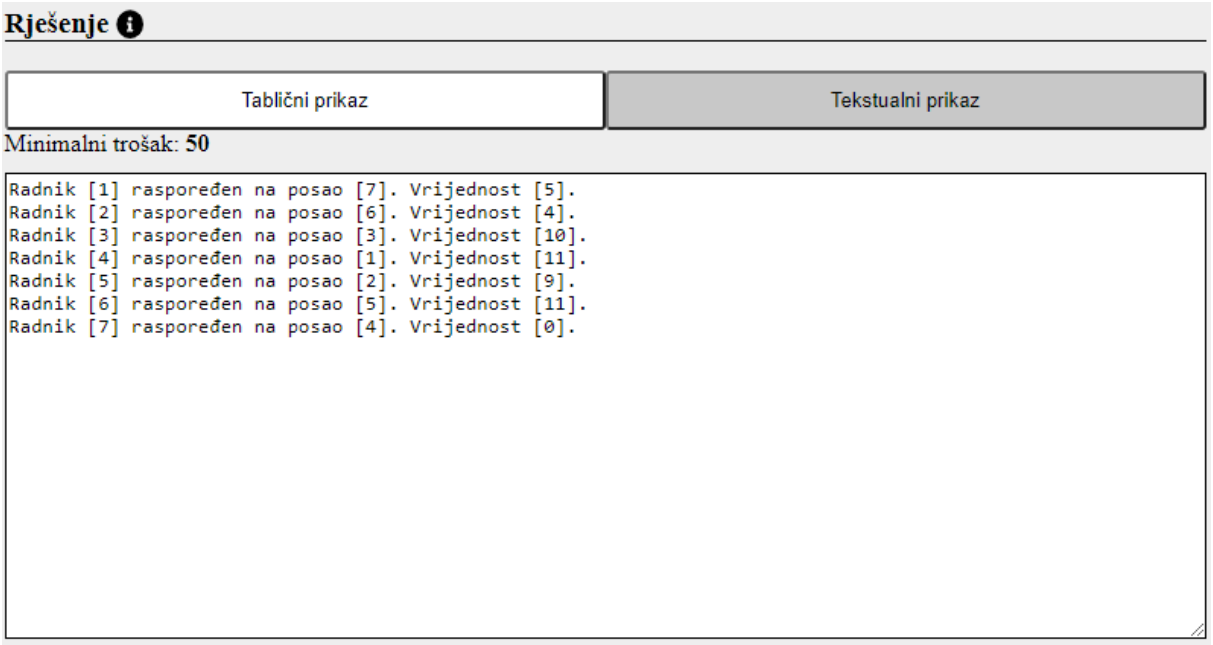
Slika 5: Primjer izgleda brzog unosa

4.2.3. Odabir algoritma

Aplikacija podržava dva algoritma tj. dvije metode za rješavanje problem raspoređivanja, mađarsku metodu i Munkres-ovu modifikaciju mađarske metode. Mađarska metoda je detaljnije opisana u poglavlju 3, a Munkres-ova modifikacija rješava problem koracima iz članka „Algorithms for the Assignment and Transportation Problems" autora James Munkres [5]. Razlozi za korištenje dva algoritma i njihove prednosti i mane više su opisane u poglavlju 4.3.

4.2.4. Rješenje

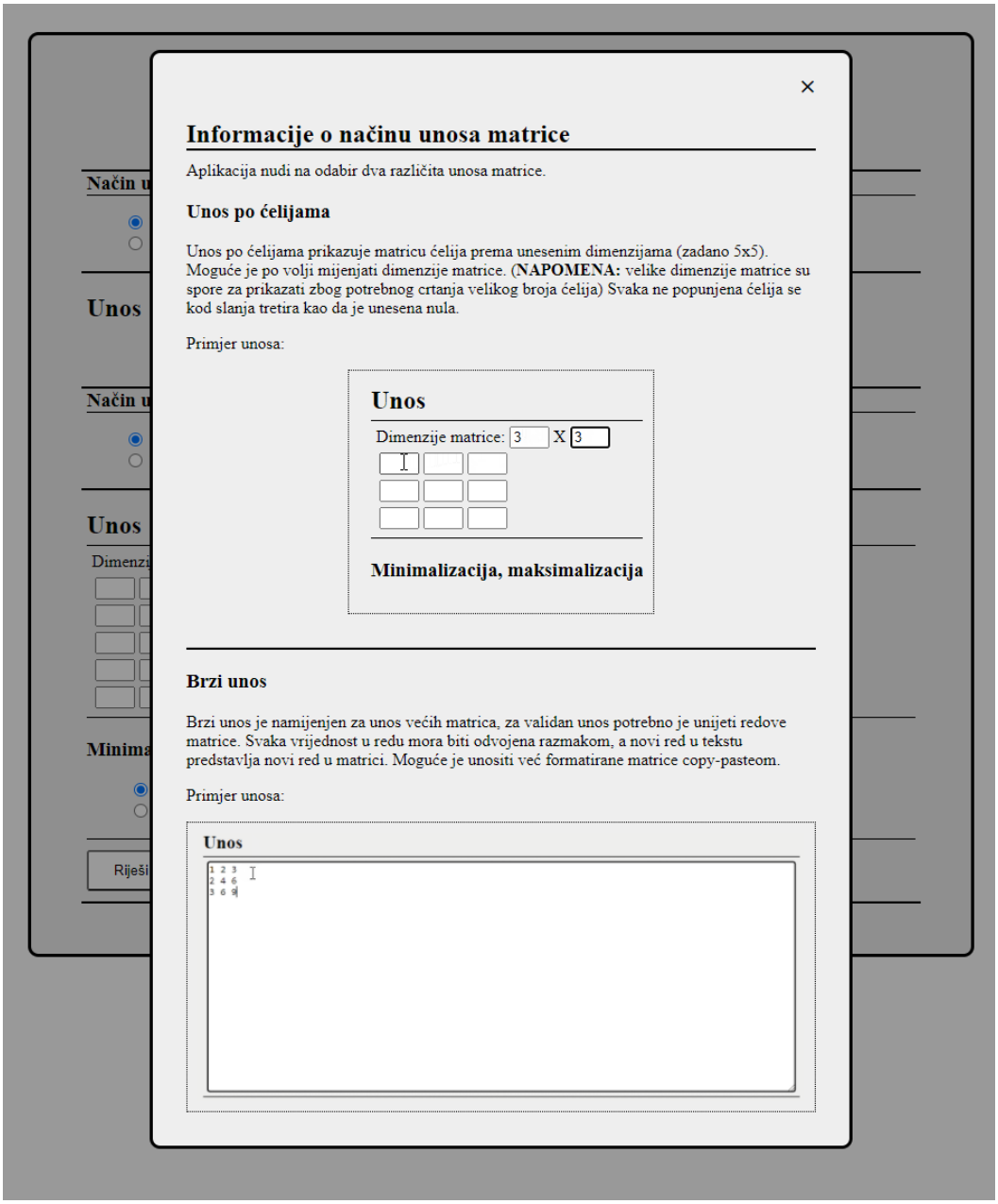
Prema unesenoj matrici, odabranom algoritmu aplikacija izračunava i prikazuje rješenje korisniku. Uvijek prikazuje minimalni trošak ili maksimalni profit, ali korisnik može konačno rješenje pregledati u tabličnom načinu prikaza ili tekstualnom prikazu. Kod tabličnog prikaza ispisana je unesena matrica, po potrebi su dodani novi redovi ili stupci ako unesena matrica nije bila kvadratna, u svakom redu i stupcu su označene odabrane vrijednosti od kojih je izračunat konačni rezultat. Tablični prikaz se može vidjeti na slici 4. Tekstualni prikaz pak ispisuje za svaki red na koji je posao radnik raspoređen te vrijednost na koju je raspoređen.



Slika 6: Primjer tekstualnog prikaza

4.2.5. Više informacija

Mnogi elementi u aplikaciji imaju postavljenu tipku za prikaz više informacija o tom elementu. Klikom na jednu od tih tipki otvara se prozor koji ukratko opisuje funkciju elementa. Slijedeća slika prikazuje prozor koji se otvara klikom na tipku za više informacija kod načina unosa matrice. U ovom primjeru prozor opisuje oba načina unosa i sadrži kratku animaciju za prikaz unosa svake vrste.



Slika 7: Primjer prozora više informacija

4.3. Problemi i izazovi u razvoju aplikacije

Kod razvoja aplikacije često se javljaju neočekivani problem i izazovi te će ovo poglavlje izdvojiti neke od specifičnih problema i izazova kod razvoja ove aplikacije.

4.3.1. Problem pronalaska optimalnog rješenja

Ovaj problem se odnosi samo na korake zadane mađarskom metodom ne na Munkres-ovu modifikaciju pošto se kod modifikacije na potpuno drugi način odabiru nule. Drugi korak mađarske metode glasi: "Potrebno je minimalnim brojem linija (horizontalnih, vertikalnih ili i jednih i drugih) precrtati sve nule u matrici reduciranih troškova. Ako je za to potrebno točno *n* linija tada moguće pronaći optimalno rješenje među precrtanim nulama, no ako je broj linija manji od *n* nastavlja se na treći korak.". U koraku nije definiran način na koji se pronalazi optimalno rješenje. te program to određuje prema navedenim koracima:

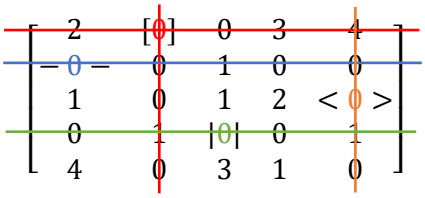
1. Pronađe red u kojemu postoji samo jedna nula i prekriži ostale u stupcu, ako nema takvog reda nastavlja na sljedeći korak.
2. Pronađe stupac koji sadrži samo jednu nulu i prekriži ostale u redu, ako nema takvog stupca nastavlja na treći korak.
3. Od svih neprekriženih nula proizvoljno se odabire jedna i prekriže se sve ostale u istom redu i stupcu.

Ovakav način odabira nula u rijetkim slučajevima ne pronalazi dobro rješenje. U slijedećem primjeru je prikazana početna matrica i iteracija matrice u kojoj je moguće pronaći optimalno rješenje jer su sve nule pokrivene s n linija, n je u ovom slučaju 5.

$$\begin{bmatrix} 2 & 0 & 1 & 4 & 4 \\ 1 & 1 & 3 & 2 & 1 \\ 2 & 1 & 3 & 4 & 1 \\ 3 & 4 & 4 & 4 & 4 \\ 4 & 0 & 4 & 2 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & [0] & 0 & 3 & 4 \\ 0 & 0 & 1 & 0 & [0] \\ 1 & 0 & 1 & 2 & 0 \\ [0] & 1 & 0 & 0 & 1 \\ 4 & 0 & 3 & 1 & 0 \end{bmatrix}$$

Ako počnemo odabirati nule prema gore navedenim koracima, prva dva koraka neće odabrati nulu te se nastavlja na treći. Treći korak odabire prvu dostupnu nulu (označena [0]) i križa sve ostale nule u istom redu i stupcu (crvena linija). Kad se prekriže potrebne nule u trećem redu ostaje samo jedna te se ta odabire (označena <0>) te se križaju sve nule u istom stupcu (narančasta linija). Treća odabrana nula se nalazi u trećem

stupcu (označena |0|) i križaju se ostale nule u istom redu (zelena linija). Zadnja nula koja se odabire je u prvom stupcu (označena -0-) i ostale nule u redu se križaju (plava linija).



Kao što vidimo iz primjera odabrane su četiri nule, no potrebno je u ovom primjeru odabrati pet i znamo da možemo odabrati točno pet jer je matrica pokrivena s točno pet linija. Ovo se događa kad uneseni problem ima više mogućih rješenja i kod odabira nula (treći korak odabira) je odabrana „kriva“ nula. Program ovaj problem pokuša riješiti na način da pokuša pronaći drugo rješenje odabirom neke druge nule kod proizvoljnog odabira.

Slijedeća slika prikazuje konačno rješenje ove matrice riješeno izrađenom aplikacijom i riješeno s dvije podržane metode. Svaka metoda je našla različit moguć raspored troškova (različit način odabira optimalnog rješenja), a krajnji rezultat je jednak.

Minimalni trošak: 7

	1.	2.	3.	4.	5.
1.	2	0	**1**	4	4
2.	1	1	3	**2**	1
3.	2	**1**	3	4	1
4.	**3**	4	4	4	4
5.	4	0	4	2	**0**

Minimalni trošak: 7

	1.	2.	3.	4.	5.
1.	2	0	**1**	4	4
2.	1	1	3	**2**	1
3.	2	1	3	4	**1**
4.	**3**	4	4	4	4
5.	4	**0**	4	2	0

Slika 8: Riješena matrica (Mađarska metoda lijevo, Munkres-ova modifikacija desno)

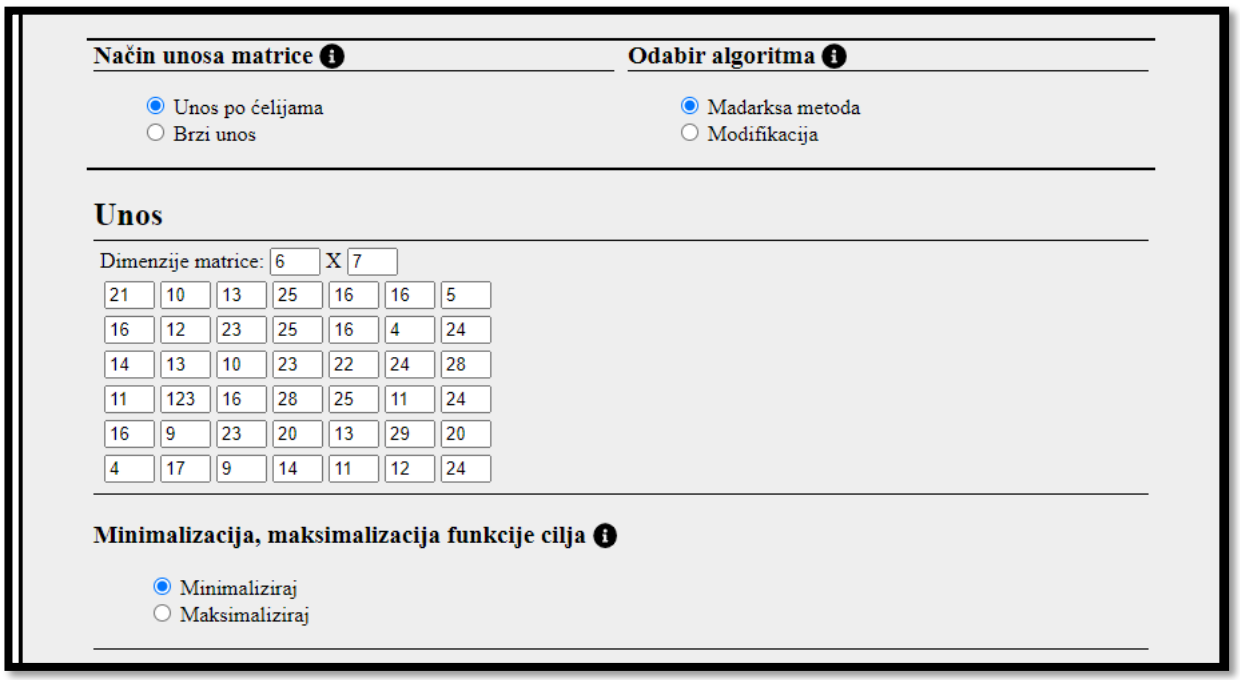
4.3.2. Munkres-ova modifikacija mađarske metode

U aplikaciju je uvedena Munkres-ova modifikacija mađarske metode kako bi se mogla provjeriti točnost rezultata i rasporeda originalne mađarske metode. James Munkres u članku [5] postavlja korake rješavanja problema raspoređivanja. Jednako kao i kod originala metoda je podijeljena na tri poduža koraka. U aplikaciji ova metoda je podijeljena na više od tri koraka.

Kako je ovo modifikacija originalne metode posjeduje neka nova svojstva. Bolja i lakša je za programsku implementaciju jer precizno definira način po kojem se odabiru nule. Također zbog preciznije definiranih koraka ova metoda ima malo bolje performanse kod rješavanja većih matrica od originala.

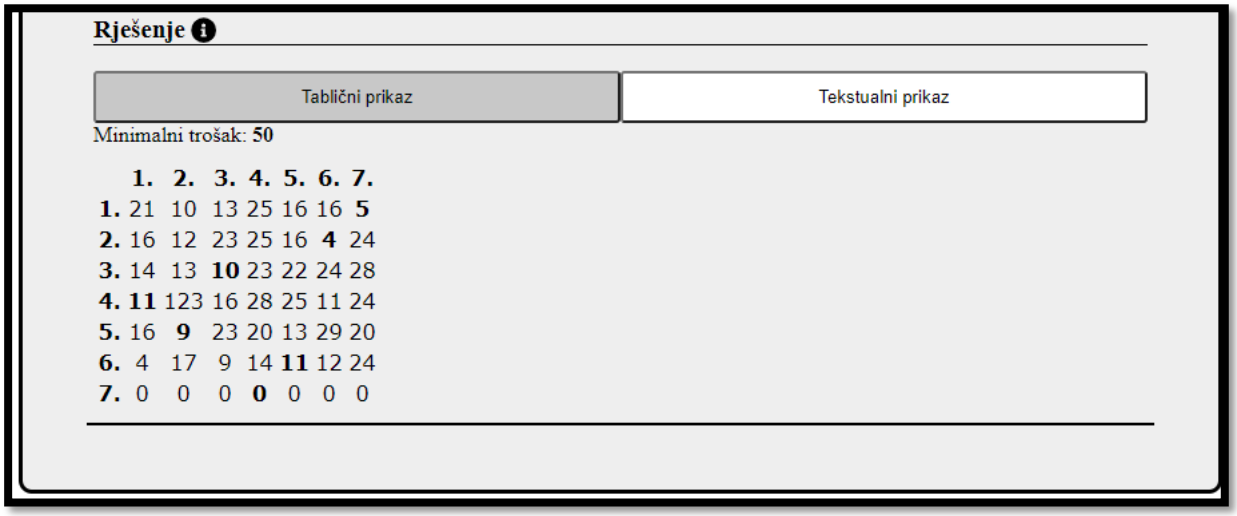
4.4. Primjer korištenja aplikacije

Primjer korištenja aplikacije koristi istu matricu zadanu u poglavlju 3.3. Prvo se odabire način unosa matrice. Kod ovog primjera odabiremo "Unos po ćelijama" zbog toga što je matrica nije prevelika i ima bolju preglednost nego "Brzi unos". Algoritam koji se odabire je mađarska metoda. Odabran je zadatak prikazan u poglavlju 3.3. te se koriste jednaki koraci rješavanja. Prije unosa matrice potrebno je postaviti potrebne dimenzije matrice. Kako se koristi već navedeni zadatak znamo da su dimenzije 6 x 7, te iste dimenzije se postavljaju u aplikaciji. Nakon postavljanja dimenzija matrice potrebno je unijeti vrijednosti iz matrice u ćelije. Zadnje što je ostalo za podesiti je funkciju cilja. Funkciju cilja se postavlja na minimalizaciju, te nakon postavljanja aplikacija izgleda kao na slijedećoj slici:



Slika 9: Postavljeni problem za rješavanje

Kad kliknemo na tipku "Riješi" aplikacija rješava unesenu matricu i prikazuje rješenje. Zbog odabira funkcije cilja za minimalizaciju aplikacija prikazuje "Minimalni trošak" i vrijednost koja u ovom slučaju iznosi 50. Ispod rezultata je prikazana unesena inicijalna matrica, unesena matrica nije bila kvadratna te ju je aplikaciju pretvorila u kvadratnu, dodala jedan red i popunila ga fiktivnim nulama. U prikazanoj matrici su označeni brojevi koji spadaju u krajnji rezultat (5 + 4 + 10 + 11 + 9 + 11 + 0 = 50) i vidi se kako su vrijednosti raspoređene.



Slika 10: Prikaz izgleda rezultata zadane matrice

5. Zaključak

Kroz rad su opisane osnove linearnog programiranja te određeni problemi linearnog programiranja. Jedan od tih problema uključuje problem raspoređivanja za koji je izrađena aplikacija kojom je taj problem moguće riješiti uz pomoć mađarske metode. Izrađena programska aplikacija bazira se na opisanoj teoriji i koracima rješavanja problema raspoređivanja mađarskom metodom navedenim u poglavljima rada koji daju prikaz teoretskih postavki mađarske metode.

Cilj ovog rada bio je prikazati i ukratko približiti problem raspoređivanja te prikazati kako se navedeni problem rješava pomoću mađarske metode. Također, opisati korake izrade aplikacije i kako izrađena programska aplikacija može poslužiti za lakše i brže rješavanje problema. Sama aplikacija kao i njezin kod aplikacije otvoreni su i dostupni svima na pregled, kopiranje, mijenjanje, korištenje, nadogradnju, te sama otvorenost koda može pružiti veliku fleksibilnost aplikacije u budućnosti. Mađarska metoda se u ovom radu pokazala kao vrlo učinkovita i brza metoda za rješavanje problema raspoređivanja u pripremljenoj programskoj aplikaciji. Kod rješavanja većih matrica s tisućama vrijednosti korištenjem mađarske metode postižu se dobre performanse i brzina kod rješavanja problema. Danas postoje metode za rješavanje problema raspoređivanja koje su brže i više pogodne za rješavanje na računalu od mađarske metode, no mađarska metoda je bila kreirana u vrijeme kada su računala bila manje dostupna, te je još uvijek jedna od najkorištenijih metoda za rješavanje problema raspoređivanja i kao što je već prije navedeno postiže solidne performanse kod rješavanja problema na računalu.

Programski kod knjižnice koja sadrži strukture i funkcije za rješavanje problema raspoređivanja pomoću mađarske metode dostupan je na sljedećoj poveznici: https://github.com/HrvojeLesar/madarska_metoda.

Kod za pokretanje web servera i grafičkog sučelja dostupan je na poveznici: https://github.com/HrvojeLesar/zavrsni_web.

Aplikacija je dostupna na korištenje na poveznici:

https://hrveklesarov.com/madarska-metoda/.

Popis literature

[1] W. Winston i J. Goldberg, *Operations Research: Applications and Algorithms*, serija Operations Research: Applications and Algorithms. Thomson Brooks/Cole, 2004., ISBN: 9780534423582. adresa: https://books.google.hr/books?id=tg5DAQAAIAAJ.

[2] D. Barković, *Operacijska istraživanja*. Osijek: Ekonomski fakultet u Osijeku, 2001., ISBN: 953-6073-51-X.

[3] C. H. Papadimitriou i K. Steiglitz, *Combinatorial optimization : algorithms and complexity*, eng. Mineola (N.Y.) : Dover, 1998., ISBN: 0486402584. adresa: http://lib.ugent.
be/catalog/rug01:000825070.

[4] Z. Lukač i L. Neralić, *Operacijska istraživanja*. Element, 2012., ISBN: 978-953-197-577-3.

[5] J. Munkres, „Algorithms for the Assignment and Transportation Problems," *Journal of the Society for Industrial and Applied Mathematics*, sv. 5, br. 1, str. 32–38, 1957., ISSN: 03684245. adresa: http://www.jstor.org/stable/2098689.

[6] R. Burkard, M. Dell'Amico i S. Martello, *Assignment Problems*, serija SIAM e-books. Society for Industrial i Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2009., ISBN: 9780898717754.

[7] A. Frieze, „An algorithm for algebraic assignment problems," *Discrete Applied Mathematics*, sv. 1, br. 4, str. 253–259, 1979., ISSN: 0166-218X. DOI: https://doi.org/10.1016/0166-218X(79)90002-7. adresa: https://www.sciencedirect.com/science/article/pii/0166218X79900027.

[8] M. Hung i W. O. Rom, „Solving the Assignment Problem by Relaxation," *Oper. Res.*, sv. 28, str. 969–982, 1980.

[9] H. W. Kuhn, „The Hungarian Method for the Assignment Problem," *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*, M. Jünger, T. M.
Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi i L. A. Wolsey, ur. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010., str. 29–47, ISBN: 9783-

540-68279-0. DOI: 10.1007/978-3-540-68279-0_2. adresa: https://doi. org/10.1007/978-3-540-68279-0_2.

[10] H. W. Kuhn, „A tale of three eras: The discovery and rediscovery of the Hungarian Method," *European Journal of Operational Research*, sv. 219, br. 3, str. 641–651, 2012., Feature

Clusters, ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2011.

11.008. adresa: http://www.sciencedirect.com/science/article/pii/ S0377221711009957.

Popis tablica

Popis slika