SVEUČILIŠTE U ZAGREBU

**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

ZAVRŠNI RAD br. 2150

**Algoritam za raspoređivanje zadataka za više robota uz pokretnu traku**

Ella Grković

Zagreb, srpanj 2025.

SVEUČILIŠTE U ZAGREBU

**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

ZAVRŠNI RAD br. 2150

**Algoritam za raspoređivanje zadataka za više robota uz pokretnu traku**

Ella Grković

Zagreb, srpanj 2025

Ovim putem želim zahvaliti svojoj mentorici na pruženoj podršci i korisnim savjetima tijekom izrade ovog završnog rada. Posebnu zahvalnost dugujem svojoj obitelji i prijateljima, čija mi je neizmjerna podrška, razumijevanje i ohrabrenje bilo ključno tijekom cijelog studija. Hvala vam što ste uvijek bili uz mene!

Sadržaj

[Uvod 1](#_Toc23263729)

[1. Naslov prvog poglavlja 2](#_Toc23263730)

[1.1. Prvo potpoglavlje 2](#_Toc23263731)

[1.2. Stilovi za tekst, naslove i podnaslove 2](#_Toc23263732)

[1.3. Stilovi za nabrajanje 2](#_Toc23263733)

[1.3.1. Stilovi za nabrajanje s točkama i crticama 3](#_Toc23263734)

[1.4. Slike 4](#_Toc23263735)

[1.5. Tablice 6](#_Toc23263736)

[1.6. Matematički izrazi 7](#_Toc23263737)

[1.7. Programski kôd 7](#_Toc23263738)

[Zaključak 9](#_Toc23263739)

[Literatura 10](#_Toc23263740)

[Sažetak 11](#_Toc23263741)

[Summary 12](#_Toc23263742)

[Skraćenice 13](#_Toc23263743)

[Privitak 14](#_Toc23263744)

# Uvod

Razvoj robotike tijekom posljednjih desetljeća ima snažan utjecaj na industriju, gospodarstvo i svakodnevni život. Roboti se danas koriste u raznim područjima – od automatizirane proizvodnje i logistike, preko medicine i istraživanja svemira, do kućne i uslužne robotike. U industrijskom kontekstu, a osobito u okviru koncepta Industrije 4.0, primjena robota postaje sve sofisticiranija, uz naglasak na autonomiju, fleksibilnost i suradnju.

Tradicionalni roboti, koji su obavljali unaprijed definirane, repetitivne zadatke u strogo kontroliranim uvjetima, danas se nadopunjuju naprednim sustavima koji mogu opažati okolinu, donositi odluke i međusobno komunicirati. Sve češće nailazimo na sustave s višestrukim robotima koji djeluju istovremeno, u kojima je ključno učinkovito upravljanje resursima, raspodjela zadataka i izbjegavanje konflikata među agentima.

Takvi sustavi traže algoritme koji mogu u realnom vremenu optimizirati ponašanje robota, uzimajući u obzir ograničenja poput vremena, prostora, fizičkog dosega i međusobnih interferencija. Jedan od tipičnih izazova u ovom području je raspoređivanje zadataka u proizvodnim sustavima u kojima objekti dolaze dinamički – primjerice, u obliku predmeta na pokretnoj traci.

Upravo takav problem promatra se u ovom završnom radu, s ciljem istraživanja i usporedbe različitih pristupa za dodjelu zadataka više robota u stvarnom vremenu.

# Opis problema

Jedan od konkretnih i čestih scenarija u industrijskoj automatizaciji uključuje robotske ruke koje preuzimaju predmete s pokretne trake. Takvi sustavi široko se primjenjuju u sortirnim centrima, pakirnicama, automobilskoj industriji, farmaceutskoj proizvodnji, kao i u modernim e-commerce skladištima.

Zajedničko svim tim sustavima je potreba da roboti brzo i precizno manipuliraju objektima koji se gibaju, bez međusobnog konflikta i uz maksimalnu iskorištenost. Sustav treba u svakom trenutku odlučiti koji robot će preuzeti koji predmet, s obzirom na njegovu poziciju, brzinu gibanja i dostupnost robota.

U ovakvom sustavu izazov je višestruk:

* Objekti dolaze dinamički – njihov broj, pozicija i vrijeme pojave nisu unaprijed poznati.
* Više robota mora djelovati istovremeno bez međusobnog preklapanja.
* Roboti imaju ograničen radni prostor – mogu dohvatiti samo dio trake.
* Vrijeme za odluku je ograničeno, jer objekt brzo napušta područje dohvatljivosti.

Problem se stoga može formulirati kao problem dodjele zadataka s vremenskim i prostornim ograničenjima, gdje je cilj:

* maksimizirati broj preuzetih objekata,
* ravnomjerno raspodijeliti opterećenje među robotima,
* izbjeći konfliktne situacije (npr. dva robota pokušavaju dohvatiti isti predmet).

Ovakav problem se može promatrati kao specifična varijanta problema fleksibilnog rasporeda poslova (Flexible Job Shop Scheduling Problem, FJSP), gdje roboti preuzimaju ulogu strojeva, a objekti su operacije koje se izvršavaju.

U ovom završnom radu razvijeni su i uspoređeni algoritmi koji rješavaju opisani problem. Prvi pristup temelji se na heurističkim metodama, gdje se koristi lokalna informacija i pravila prioriteta za donošenje odluka. Drugi pristup koristi formalni optimizacijski model temeljen na programiranju s ograničenjima (Constraint Programming) pomoću Google OR-Tools biblioteke.

Analiza rezultata provest će se na temelju simuliranih podataka, kroz usporedbu efikasnosti (broj preuzetih paketa), robusnosti i vremenske složenosti. Time se želi dobiti uvid u praktične razlike između jednostavnih i naprednih pristupa raspodjeli zadataka u višerobotskim sustavima.

.

# Fleksibilni raspored poslova (Flexible Job Shop Scheduling Problem)

Problem rasporeda poslova (Job Shop Scheduling Problem – JSSP) jedan je od temeljnih problema u teoriji rasporeda i operacijskom istraživanju. Klasični oblik uključuje skup poslova, gdje se svaki sastoji od fiksnog slijeda operacija koje se moraju izvršiti na točno određenim strojevima. Svaki stroj može istovremeno obraditi najviše jednu operaciju, a cilj je pronaći raspored koji minimizira ukupno vrijeme trajanja (makespan) ili neki drugi kriterij.

U fleksibilnoj varijanti, poznatoj kao Flexible Job Shop Scheduling Problem (FJSP), uvodi se dodatna dimenzija slobode: svaka operacija može biti izvršena na jednom od više mogućih strojeva. Ovo proširenje znatno povećava složenost problema, ali i omogućuje realističnije modeliranje mnogih stvarnih sustava.

## Povezanost s višerobotskim sustavima

Problem dodjele zadataka robotima uz pokretnu traku u ovom radu može se promatrati kao posebna varijanta FJSP-a, gdje:

* Paketi predstavljaju operacije koje treba izvršiti,
* Roboti predstavljaju strojeve koji mogu izvršavati operacije,
* Svaki robot ima ograničen radni prostor i vrijeme, što uvodi dodatna ograničenja u usporedbi s klasičnim JSSP-om,
* Cilj nije minimizacija vremena već maksimizacija broja obavljenih zadataka (što je varijanta ciljne funkcije u FJSP-u).

## Važnost FJSP modela

Model FJSP koristi se u:

* Pametnim tvornicama (smart manufacturing),
* Automatiziranim skladištima i sortirnicama,
* Višerobotskim koordinacijama u stvarnom vremenu.

Primjenom ovog modela moguće je koristiti napredne metode rješavanja kao što su programiranje s ograničenjima, metaheuristike i kombinatorna optimizacija, što je upravo pristup korišten u ovom radu.

# Korištene tehnologije

## Google OR-Tools

GoogleOR-Tools je moćna, besplatna i open-source biblioteka za rješavanje problema iz područja operacijskog istraživanja (Operations Research). Razvijena je od strane inženjera u Googleu kako bi se nosila s raznim izazovima rasporeda, rutiranja, optimizacije resursa i slično – u velikim, stvarnim sustavima poput onih koji se koriste u logistici, distribuciji i industrijskoj automatizaciji.

OR-Tools podržava nekoliko paradigmi rješavanja, uključujući:

* CP-SAT – moderni solver za programiranje s ograničenjima koji koristi kombinaciju propagacije, grananja i naprednih heuristika.
* Linearno i cjelobrojno programiranje (LP/MIP) putem popularnih rješavača kao što su GLOP, SCIP i Glpk.
* Probleme rutiranja (Routing Solver) – posebno dizajniran za vozila, dostavne rute, planiranje putanja i sl.

### CP-SAT rješavač (Constraint Programming - Satisfiability)

CP-SAT je glavni rješavač unutar Google OR-Tools biblioteke za rješavanje problema programiranja s ograničenjima (eng. *Constraint Programming*, CP). Naziv "SAT" dolazi od činjenice da se problemi transformiraju u varijantu problema zadovoljivosti booleovih formula (SAT), čime se spaja snaga logičkog zaključivanja i kombinatoričke optimizacije.

**Ključne značajke CP-SAT rješavača:**

* **Podrška za booleove i cjelobrojne varijable** – omogućuje modeliranje složenih uvjeta koji uključuju logičke odluke, vremenske ovisnosti i više-agentske interakcije (npr. više robota).
* **Učinkovita propagacija i domenska redukcija** – CP-SAT koristi napredne tehnike za eliminaciju nemogućih vrijednosti što značajno smanjuje veličinu pretražnog prostora.
* **Podrška za uvjetna ograničenja (reified constraints)** – npr. OnlyEnforceIf(z[i]) omogućuje da se neko pravilo aktivira samo ako je zadatak aktivan, što je iznimno korisno u višerobotskim i dinamičkim sustavima.
* **Kombinacija CP + SAT + Integer Linear Programming** – CP-SAT je hibridni rješavač koji kombinira snagu različitih metoda: propagatora iz CP-a, zaključivanja iz SAT-a i presjeka domena iz ILP-a.
* **Skalabilnost** – iako je orijentiran prema "hard" kombinatoričkim problemima, CP-SAT je vrlo skalabilan i sposoban rješavati probleme s tisućama varijabli i ograničenja.

## Python

Python je programski jezik visokog nivoa, opće namjene, poznat po jednostavnoj sintaksi, čitljivosti i izrazitoj svestranosti. Nastao je početkom 1990-ih godina, a njegova popularnost kontinuirano raste zahvaljujući jednostavnosti korištenja te širokom spektru primjena koje uključuju razvoj softvera, web aplikacije, znanstvena istraživanja, automatizaciju, strojno učenje i brojne druge oblasti.

Jedna od glavnih karakteristika Pythona je njegova podrška različitim paradigmama programiranja, kao što su proceduralno, objektno-orijentirano i funkcionalno programiranje, čime omogućava fleksibilan pristup razvoju različitih vrsta aplikacija. Uz to, Python ima bogatu standardnu biblioteku koja olakšava brojne zadatke poput obrade teksta, rada s datotekama, mrežnog programiranja te pristupa bazama podataka.

Posebno je značajno područje primjene Python jezika analiza podataka i znanstveno računarstvo. Biblioteke poput NumPy (numerički izračuni), pandas (manipulacija podacima), matplotlib (vizualizacija podataka) i SciPy (znanstveni izračuni) pružaju moćne alate za manipulaciju, analizu i vizualizaciju podataka, što je ključno u znanosti, istraživanjima i industriji.

Za potrebe ovog završnog rada Python je osobito pogodan zbog svoje izvrsne podrške za optimizacijske probleme putem Google OR-Tools biblioteke. OR-Tools omogućuje učinkovito modeliranje i rješavanje složenih optimizacijskih zadataka kao što su problemi kombinatorne optimizacije, linearno programiranje i programiranje s ograničenjima. Time Python pruža praktično i efikasno okruženje za razvoj i analizu algoritama korištenih u ovom radu, osiguravajući pouzdana i primjenjiva rješenja.

# Optimizacijski problem

## Opis modela

Zadan je skup zadataka , pri čemu svaki zadatak odgovara jednom predmetu na traci. Pozicija predmeta iu trenutku razmaranja je (). Zadan je skup robota . Za robota definiraju se granice njegovog radnog prostora i . Također, definira se i za svakog robota njegova pozicija ( ,). Brzine trake je , a koordinatni sustav postavljen je prema slici

A graph with red and yellow dots

Description automatically generated

Slika 5.1. Geometrijski prikaz sustava

## Vremenski parametri

Sustav se razmatra za trenutak . Definiraju se sljedeće vrijednosti:

- vrijeme kad posao postaje dostupan robotu (engl. release time). Ta vrijednost određena je radnim prostorom robota i pozicijom predmeta na traci.

- vrijeme kad posao postaje nedostupan robotu (engl. deadline time), jer izlazi iz njegovog radnog prostota.

- trenutak u kojem robot uzima predmet . Ako robot uzima predmet, vrijedi:

- vrijeme obrade posla na stroju , koje se sastoji od uzimanja predmeta i odlaganja predmeta. Ova vrijednost ovisi o položaju predmeta u trenutku kad se uzima s trake. Neka je vrijeme u kojem robot uzima predmet s trake. Ako je u trenutku predmet bio na poziciji (), pozicija predmeta u trenutku je a se može izračunati kao:

Aproksimirana vrijednost je:

## Varijable optimizacije

- ∈ {0, 1}: pokazuje je li paket preuzet

- ∈ {0, 1}: označava je li zadatak dodijeljen robotu

- ∈ ℤ: vrijeme preuzimanja paketa

- ∈ {0, 1}: određuje redoslijed izvršavanja zadataka i na istom robotu

## Ciljna funkcija

Cilj je maksimizirati broj preuzetih paketa:

## Ograničenja

Cilj se ostvaruje uz sljedeća ograničenja koja vrijede za svako rješenje ( je oznaka za jako veliki broj):

1. Svaki zadatak koji se izvršava dodijeljen je najviše jednom robotu:

2. ako se zadaci izvršavaju (ako je i) i ako su dodijeljeni istom robotu (ako je i ), pazi da se ne izvršavaju istovremeno tj. da nema preklapanja:

3. Vrijeme dohvaćanja ovisi o poziciji paketa:

## Primjer rješenja

Neka je zadano da u sustavu postoje dva robota . Neka u sustavu postoji pet zadataka . Ako se npr. predmet 5 preskače, a prva dva predmeta radi robot 1, a druga dva robot 2, rješenje će biti:

Ako se na robotu 1 prvo izvršava zadatak 1 pa zadatak 2, a na robotu prvo predmet 4, pa 3, vrijedit će:

Trebaju također biti određene vrijednosti koje su neki cijeli brojevi. Vrijednost jer se predmet ne uzima s trake. Rješenje koje nije moguće prekršit će barem jedan od napisanih uvjeta tj. ograničenja.

# Implementacija rješenja

Implementacija rješenja izrađena je u programskom jeziku Python, korištenjem biblioteke Google OR-Tools, točnije CP-SAT rješavača koji omogućuje modeliranje i rješavanje problema programiranja s ograničenjima (eng. Constraint Programming).

U nastavku je opisana struktura programa, ključne varijable i funkcionalnosti, uz komentare i obrazloženja pojedinih implementacijskih odluka.

## Inicijalizacija sustava

Na početku implementacije definiraju se ulazni parametri:

v = 1

robots = [(5, 10), (15, 10)] # pozicije robota

workspaces = [(10, 0), (20, 10)] # dometi robota

T\_MAX = 50

H = 1000 # velika konstanta

Zatim se korisnika traži unos broja paketa, te se njihove početne pozicije generiraju nasumično unutar granica x ∈ [20, 50] i y ∈ [0, 10]:

num\_items = int(input("Unesi broj paketa: "))

items = []

for i in range(num\_items):

x = random.randint(20, 50)

y = random.randint(0, 10)

items.append((x, y))

Na kraju se ispisuju generirane koordinate svakog paketa:

for i, item in enumerate(items):

print(f"Paket{i+1} : ({item[0]}, {item[1]})")

## Postavljanje modela i varijabli

Na početku implementacije definira se instanca CpModel, što je osnovni objekt za definiranje ograničenja i ciljne funkcije:

model = cp\_model.CpModel()

Zatim se definiraju glavne varijable:

* z[i]: binarne varijable koje označavaju je li paket i preuzet.
* alpha[k][i]: binarne varijable koje označavaju je li paket i dodijeljen robotu k
* t[i]: cjelobrojne varijable koje predstavljaju trenutak preuzimanja paketa
* beta[i][j]: binarne varijable za određivanje redoslijeda zadataka na istom robotu
* Tk[k][i] – trajanje dohvaćanja paketa i robotom k

Varijable su kreirane korištenjem NewBoolVar i NewIntVar, uz pažljivo zadane granice:

# z\_i - indikator hoće li se paket i uzeti

for i in range(num\_items):

z.append(model.NewBoolVar(f'z[{i}]'))

# t\_i - cjelobrojno vrijeme početka dohvaćanja paketa i

for i in range(num\_items):

t.append(model.NewIntVar(0, T\_MAX, f't[{i}]'))

# alpha\_{k,i} - je li paket i dodijeljen robotu k

for k in range(num\_robots):

row = []

for i in range(num\_items):

row.append(model.NewBoolVar(f'alpha[{k}][{i}]'))

alpha.append(row)

# beta\_{i,j} - je li i prije j na istom robotu

for i in range(num\_items):

row = []

for j in range(num\_items):

row.append(model.NewBoolVar(f'beta[{i}][{j}]'))

beta.append(row)

#Tk\_{i,k} - vrijeme obrade posla i na stroju k

for k in range(num\_robots):

row = []

for i in range(num\_items):

var = model.NewIntVar(0, H, f'Tk[{k}][{i}]')

row.append(var)

Tk.append(row)

## Ograničenja dodjele

Dodjela paketa robotima

Prvo važno ograničenje je da svaki paket može biti dodijeljen najviše jednom robotu:

for i in range(num\_items):

model.Add(sum(alpha[k][i] for k in range(num\_robots)) == z[i])

Ako je z[i] = 1, tada je barem jedan alpha[k][i] = 1. Ako je z[i] = 0, tada nijedan robot ne smije dohvatiti taj paket.

Računanje trajanja dohvaćanja

Za svaki par (robot, paket), izračunava se pozicija paketa u trenutku dohvaćanja:

x\_pos = xi - t[i] \* v

Zatim se računa horizontalna udaljenost od robota:

abs\_dx = |xk - x\_pos|

Vertikalna udaljenost |yk - yi| se računa direktno jer je yk i yi konstanta. Ukupno trajanje dohvaćanja modelira se kao:

Tk[k][i] = abs\_dx + dy

Vremenski prozor dostupnosti paketa

Paket se može dohvatiti samo dok se nalazi unutar radnog prostora robota:

tr = max(0, int((xi - Wmin) / v))

td = max(0, int((xi - Wmax) / v))

model.Add(t[i] >= tr).OnlyEnforceIf(alpha[k][i])

model.Add(t[i] <= td).OnlyEnforceIf(alpha[k][i])

Ovo je implementirano putem OnlyEnforceIf, kako bi se uvjeti aktivirali samo ako je paket stvarno dodijeljen robotu k.

Izbjegavanje preklapanja

Za svaki par paketa (i, j) i svakog robota k, koristi se tzv. diskretna disjunkcija za izbjegavanje preklapanja:

model.Add(t[i] >= t[j] + Tk[k][j] - H \* (4 - z[i] - z[j] - alpha[k][i] - alpha[k][j] + beta[i][j]))

model.Add(t[j] >= t[i] + Tk[k][i] - H \* (5 - z[i] - z[j] - alpha[k][i] - alpha[k][j] - beta[i][j]))

Ove dvije linije zajedno osiguravaju da se zadaci i i j ne preklapaju ako ih isti robot dohvaća. Koristi se velika konstanta H da se izrazi uvjetna aktivacija, ovisno o stanju varijabli z, alpha i beta.

## Ciljna funkcija

Cilj optimizacije je maksimizirati broj uspješno preuzetih paketa:

model.Maximize(sum(z))

Time se model vodi prema rješenju gdje se obavi što više zadataka, ali unutar definiranih vremenskih i prostorno-vremenskih ograničenja.

## Rješavanje i ispis rješenja

Model se rješava CP-SAT rješavačem:

solver = cp\_model.CpSolver()

status = solver.Solve(model)

Ako rješenje postoji, ispisuju se informacije o svakom paketu:

* je li preuzet (z[i])
* u kojem trenutku (t[i])
* koji robot ga je preuzeo (alpha[k][i])
* koliko traje dohvat (Tk[k][i])

A screenshot of a computer

Description automatically generated

Slika 5.1. Prikaz ispisa

## Vizualizacija rasporeda

Za lakšu interpretaciju rješenja izrađuje se **Ganttov dijagram** pomoću matplotlib. Svaki paket koji je preuzet prikazuje se kao obojana traka na vremenskoj osi za pripadajućeg robota. Paket je označen oznakom P1, P2, itd.

ax.broken\_barh([(start, duration)], (robot\_index, height), ...)

ax.text(..., f"P{item+1}", ...)

A diagram of different colored squares

Description automatically generatedSlika 5.2. Primjer Ganttovog dijagrama

Na ovaj način moguće je intuitivno vizualizirati redoslijed preuzimanja i stupanj iskorištenosti svakog robota.

## Heuristički algoritam – pohlepna strategija

Osim optimalnog modela temeljenog na CP-SAT pristupu, implementirana je i pohlepna heuristika koja koristi jednostavan, efikasan pristup za dodjelu paketa robotima. Cilj heurističkog rješenja je omogućiti brzo rješenje u situacijama kada optimizacija nije vremenski isplativa ili kada je potrebno pronaći rješenje u realnom vremenu.

Logika algoritma

Algoritam iterira kroz sve pakete sortirane po koordinati x (odnosno, po redoslijedu pristizanja s trake). Za svaki paket se određuje je li u radnom području nekog robota i dodjeljuje se onom koji ga može prvi dohvatiti.

for i, (x, y) in enumerate(items):

for k in range(num\_robots):

xk, yk = robots[k]

Wmin, Wmax = workspaces[k]

if Wmin <= x <= Wmax:

dx = abs(xk - x)

dy = abs(yk - y)

trajanje = dx + dy

...

Ako se paket nalazi unutar radnog prostora robota k, dohvat se dodjeljuje tom robotu ako je trenutno slobodan ili ako ne postoji konflikt s prethodnim paketima.

# Analiza rezultata

## Usporedba optimalnog rješenja sa rješenjima pohlepnog algoritmom

U sklopu ovog rada implementirana su dva pristupa za dodjelu zadataka robotima koji preuzimaju pakete s pokretne trake: pohlepan algoritam, koji u svakom trenutku donosi lokalno optimalnu odluku, te optimalni algoritam, koji koristi optimizacijski pristup s ciljem maksimizacije ukupnog broja preuzetih paketa.

Evaluacija algoritama provedena je na unaprijed generiranim skupovima podataka koji sadrže 5, 10 i 15 paketa po testnom skupu. Za svaki broj paketa izvršeno je više pokretanja algoritama na različitim instancama, a rezultati su uspoređeni prema broju preuzetih paketa. Rezultati su prikazani u obliku boxplot dijagrama (Slika 1.1), te sažeti u tablici u nastavku.

Tablica 1.1 prikazuje usporedbu učinkovitosti između pohlepnog i optimalnog algoritma, gdje je jasno vidljivo da optimalni pristup ostvaruje bolji rezultat u svim slučajevima – osobito kod većeg broja paketa.

Iz dobivenih rezultata vidljivo je da optimalni algoritam nadmašuje pohlepan pristup u svim slučajevima. Za skupove od 5 paketa razlika je minimalna jer su svi paketi preuzeti. Međutim, kako broj paketa raste, prednost optimalnog algoritma postaje sve izraženija. Pohlepan algoritam pokazuje veću varijabilnost rezultata i često ne uspijeva ostvariti maksimalan broj preuzetih paketa.

Optimalni algoritam dosljedno ostvaruje rezultate blizu maksimuma, čime potvrđuje svoju učinkovitost u složenijim scenarijima. Međutim, zbog veće računalne složenosti njegova primjena može biti vremenski zahtjevnija, osobito kod većeg broja paketa i robota. S druge strane, pohlepan algoritam, iako jednostavniji i brži, postiže slabiju ukupnu učinkovitost.

Tablica 1.1. Usporedba rezultata

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Broj paketa | Algoritam | Prosječan broj preuzetih paketa | Medijan | Minimalno | Maksimalno |
| 5 | Pohlepan | 4,5 | 5 | 3 | 5 |
| 5 | Optimalni | 5,0 | 5 | 5 | 5 |
| 10 | Pohlepan | 7,2 | 7 | 4 | 9 |
| 10 | Optimalni | 9,5 | 10 | 8 | 10 |
| 15 | Pohlepan | 9,1 | 9 | 7 | 11 |
| 15 | Optimalni | 12,6 | 13 | 10 | 14 |

A graph with different colored squares

Description automatically generated

Slika 1.1. Dijagram usporedbe

Na boxplot dijagramu (Slika 1.1) prikazane su raspodjele rezultata za pohlepni i optimalni algoritam pri različitom broju paketa. Uočljivo je da:

Optimalni algoritam daje vrlo konzistentne rezultate – svi izlazi su blizu maksimuma, a kutije (interkvartilni raspon) su uske ili potpuno stisnute, što ukazuje na nisku varijabilnost i visoku pouzdanost algoritma.

Pohlepni algoritam, s druge strane, pokazuje veću disperziju rezultata, osobito kod 10 i 15 paketa. Pojavljuju se slučajevi gdje broj preuzetih paketa znatno odstupa od medijana, što znači da algoritam često ovisi o konkretnoj instanci zadatka.

Kod 5 paketa razlike su manje izražene jer je problem dovoljno jednostavan da i pohlepan pristup često uspije pokupiti sve pakete. Kod 10 i 15 paketa vidljiv je jasniji kontrast – optimalni algoritam uvijek iskorištava resurse do maksimuma, dok pohlepni može promašiti do 4–5 paketa.

Brkovi (eng. whiskers) u boxplotu za pohlepni algoritam pokazuju prisutnost ekstremnih slučajeva gdje je izvedba izrazito loša, dok optimalni algoritam takvih odstupanja nema.

## Usporedba optimalnih rješenja s obzirom na položaj robota

Za dodatnu evaluaciju optimalnog algoritma provedeno je ispitivanje njegove osjetljivosti na početne položaje robota. Cilj je bio utvrditi koliko raspored robota u prostoru utječe na ukupnu učinkovitost preuzimanja paketa.

U tu svrhu testirani su identični skupovi podataka kao u prethodnoj analizi (5, 10 i 15 paketa), ali su pozicije robota bile postavljene u obrnutom redoslijedu u odnosu na originalnu konfiguraciju – svaki robot sa jedne strane pokretne trake u odnosu na početan položaj u kojem su oba robota bila sa iste strane trake.

Rezultati su prikazani u obliku boxplot dijagrama (Slika 1.2), gdje su uspoređene vrijednosti za originalnu i obrnuto postavljenu konfiguraciju robota za svaki testni slučaj.

Tablica 1.2 pokazuje da položaj robota ima značajan utjecaj na broj uspješno preuzetih paketa, osobito kod većeg broja paketa. Kod 5 paketa razlike nisu značajne jer je broj mogućih kombinacija mali i jednostavan za optimizaciju. Međutim, kod 15 paketa, prosjek u obrnutoj konfiguraciji nadmašuje standardnu, što ukazuje da u određenim situacijama i takvi rasporedi mogu rezultirati boljom koordinacijom.

Ovi rezultati ukazuju na važnost pažljivog dizajna početnih konfiguracija u stvarnim sustavima, ali i na robusnost algoritma u smislu prilagodbe na različite rasporede robota.

Tablica 1.2. Usporedba rezultata

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Broj paketa | Konfiguracija | Prosječan broj preuzetih paketa | Medijan | Minimalno | Maksimalno |
| 5 | Optimalni | 5,0 | 5 | 5 | 5 |
| 5 | Optimalni – obrnuti | 5,0 | 5 | 5 | 5 |
| 10 | Optimalni | 9,5 | 10 | 8 | 10 |
| 10 | Optimalni – obrnuti | 10,0 | 10 | 10 | 10 |
| 15 | Optimalni | 12,6 | 13 | 10 | 14 |
| 15 | Optimalni – obrnuti | 14,1 | 14 | 13 | 15 |

A graph with different colored squares

Description automatically generated with medium confidence

Slika 1.2. Dijagram usporedbe

Boxplot dijagram (Slika 1.2) prikazuje utjecaj početnog rasporeda robota na učinkovitost preuzimanja paketa.

Kod 5 paketa razlike gotovo da nema – oba rasporeda rezultiraju identičnim rasponima i centralnim vrijednostima.

Kod 10 paketa, obrnut raspored robota postiže čak nešto bolje rezultate – svi rezultati su maksimalni, bez varijabilnosti. To sugerira da simetrična raspodjela robota uz traku može rezultirati boljom koordinacijom i raspodjelom posla.

Kod 15 paketa ta prednost postaje još izraženija: medijan i prosjek su viši, a varijabilnost manja. To ukazuje da raspored robota može imati značajan utjecaj kod većih opterećenja i složenijih konfiguracija.

Boxplotovi za obrnuti raspored pokazuju uži interkvartilni raspon i kraće brkove, što znači da je sustav stabilniji u izvedbi.

Ovi rezultati potvrđuju važnost pažljivog dizajna rasporeda robota u stvarnim industrijskim primjenama, osobito kad je zadatkovno opterećenje visoko.

# Zaključak

U ovom završnom radu istražen je problem dodjele zadataka više robota u sustavu s pokretnom trakom, što je tipičan scenarij u suvremenoj industrijskoj automatizaciji. Cilj je bio razviti i usporediti algoritme koji dodjeljuju pakete robotima na način koji maksimizira učinkovitost sustava, mjerenu kroz broj uspješno preuzetih paketa.

Implementirana su dva pristupa: pohlepni algoritam temeljen na heuristici, koji donosi lokalno optimalne odluke, i optimizacijski algoritam zasnovan na programiranju s ograničenjima (CP-SAT), koji globalno maksimizira broj preuzetih paketa uz poštivanje prostorno-vremenskih ograničenja.

Eksperimentalna analiza pokazala je da optimizacijski pristup daje značajno bolje rezultate, osobito u složenijim scenarijima s većim brojem paketa. Iako je pohlepni algoritam brži i jednostavniji, njegova učinkovitost varira ovisno o rasporedu paketa i robotima. Također je pokazano da raspored robota u prostoru može značajno utjecati na konačnu učinkovitost sustava, što potvrđuje važnost pažljive konfiguracije u praksi.

Dobiveni rezultati potvrđuju da primjena formalnih optimizacijskih metoda poput CP-SAT-a omogućuje pouzdano i učinkovito upravljanje višerobotnim sustavima, te otvaraju prostor za daljnji razvoj i proširenje modela, primjerice uvođenjem prioriteta, vremenskih prozora ili energetske optimizacije.

# Literatura

1. Google OR-Tools. Google Developers. Poveznica: https://developers.google.com/optimization
2. Google Charts Documentation – Boxplot. Google Developers. Poveznica: <https://developers.google.com/chart/interactive/docs/gallery/boxplot>
3. Jia, Y., Tang, H., Yang, B., Wang, Q. Multi-robot task assignment and scheduling for moving targets with time windows. European Journal of Operational Research, 311(2) (2023), str. 778–793. Poveznica: <https://www.sciencedirect.com/science/article/pii/S037722172300382X>
4. Wikipedia. Job-shop scheduling. Poveznica: <https://en.wikipedia.org/wiki/Job-shop_scheduling>
5. Plotly Technologies Inc. Creating Gantt Charts in Python. Poveznica: <https://plotly.com/python/gantt/>
6. Tang, H., Zhang, Y., Zhang, W., & Tian, Y. A task allocation framework for multi-robot systems targeting moving objects. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Montreal, Canada (2019), str. 6153–6159. Poveznica: <https://par.nsf.gov/servlets/purl/10149453>
7. Python Software Foundation. Python Documentation. Dostupno na: <https://www.python.org/doc/>

# Sažetak

Naslov:

Algoritam za raspoređivanje zadataka za više robota uz pokretnu traku

Sadržaj:

Robotske ruke koje skupljaju predmete na pokretnoj traci koriste se u brojnim industrijskim primjenama. Ovisno o položaju objekata i robota, potrebno je robotima dodijeliti objekte tijekom rada sustava kako bi se postigli što bolji parametri učinkovitosti kao što su broj pokupljenih predmeta, iskorištenost robota itd. U ovom završnom radu potrebno je ovaj zadatak opisati kao problem optimizacije dodjele zadataka. Potrebno je istražiti algoritme za rješavanje navedenog optimizacijskog problema, odabrati dva algoritma za njegovo rješavanje i implementirati ih u programski jezik Python. Primjeri algoritama su dinamičko programiranje, dodjela po pravilima prioriteta i slični. Potrebno je analizirati učinkovitost rješenja za više ulaznih podataka.

Ključne riječi:

roboti, Python, optimizacijski algoritam, paketi, Or-Tools

# Summary

Title:

Task Allocation Algorithm for Multiple Robots Along a Conveyor Belt

Summary:

Robotic arms that pick up items from a conveyor belt are widely used in various industrial applications. Depending on the positions of objects and robots, it is necessary to assign objects to robots during system operation in order to achieve optimal performance indicators, such as the number of items successfully picked up, robot utilization, and overall system efficiency. This thesis addresses the problem by formulating it as a task allocation optimization problem. Several algorithms for solving this type of problem are explored, and two selected approaches are implemented in the Python programming language. Example algorithms include dynamic programming, greedy assignment strategies, and similar methods.

The effectiveness of the implemented algorithms is analyzed using different input datasets to assess their performance under various conditions.

Keywords:

robots, Python, optimization algorithm, packages, OR-Tools