SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI PROJEKT

Raspoređivanje zadataka u više-robotskom sustavu korištenjem metode aukcije

Filip Škoro

Zagreb, lipanj 2021.

Završni projekt [183456]

Student/ica	Filip Škoro
e-pošta	filip.skoro@fer.hr
Mentor/ica	prof. dr. sc. Stjepan Bogdan
Jezik	Hrvatski
Naslov (na hrvatskom)	Raspoređivanje zadataka u više-robotskom sustavu korištenjem metode aukcije
Naslov (na engleskom)	Task allocation in multi-robot systems based on auction method
Zadatak (na hrvatskom)	U svrhu što efikasnijeg prikupljanja podataka sa senzora raspoređenih u nekom prostoru korištenjem više-robotskog sustava, potrebno je razviti algoritam za raspoređivanje zadataka temeljen na iterativnoj aukciji s prioritetima u programskom jeziku Python. Prilikom određivanja optimalnog rasporeda, algoritam mora u obzir uzeti različite sposobnosti robota te vremenska i sinkronizacijska ograničenja. Kako bi se odredila efikasnost ovog pristupa, potrebno je generirati set nasumičnih instanci opisanog problema te izvršiti planove u postojećem 2D simulatoru.

Zahvaljujem se svom mentoru prof. dr. sc. Stjepanu Bogdanu na izvrsnom prijedlogu teme za završni projekt te na korisnim savjetima i pruženim materijalima.

Zahvaljujem se i docentici doc. dr. sc. Tamari Petrović te asistentu Marku Križmančiću, mag. ing. što su pratili moj rad na projektu te mi savjetima i znanjem pomogli u izradi projekta.

Sadržaj

1. Uvod

U današnjem modernom odnosno digitalnom svijetu nerijetka je pojava da veliki sustavi; tvrtke, organizacije, skladišta, bolnice, tvornice i druga postrojenja koriste robote ili neke druge automatizirane uređaje za obavljanje zadataka i poslova. U svrhu što veće efikasnosti, takvi sustavi često puta nailaze na probleme oko organizacije i podjele poslova. Zadnjih godina sve je više i više studija, istraživanja i znanstvenih članka vezanih za razvoj ideje optimalne podjele poslova u više-robotskim sustavima uzevši u obzir mogućnosti robota kao i prioritete poslova te želje proizvođača i korisnika [3], [4], [5] itd.

U ovome završnom radu objašnjena je jedna od metoda koja nudi rješenje problema podjele zadataka u više-robotskom sustavu u svrhu stvaranja optimalnog rasporeda - metoda aukcije. Potreba za razvijanjem ovakve metode nalazi se u tome što je općenito metode koje se baziraju na aukciji moguće ostvariti tako da budu decentralizirane, a pritom fleksibilne u odnosu na druge ciljeve i zadatke [4]. Također, još jedna od motivacija za razvoj i primjenu metode aukcije u više-robotskim sustavima je potreba za usporedbom s drugim metodama koje nude svoje rješenje optimalnog rasporeda u svrhu dobivanja rezultata i odgovora na pitanje koja metoda pruža optimalno rješenje.

Ovaj završni rad strukturiran je na sljedeći način: u poglavlju Definicija i opis zadatka ukratko je predstavljen problem za koji tražimo rješenje odnosno zadatak koji mora biti izvršen metodom koja se obrađuje u ovom radu. U poglavlju Metoda aukcije opisani su i objašnjeni algoritmi koji čine metodu aukcije. Nakon toga slijedi poglavlje Implementacija gdje je ukratko rečeno kako je metoda programski izvedena, koje funkcije obavljaju zadatke algoritama objašnjenih u prethodnom poglavlju, jesu li neki algoritmi drugačije izvedeni itd. Na kraju imamo poglavlje Rezultati i primjeri gdje je grafički prikazano i objašnjeno nekoliko primjera koji bi trebali potvrditi ispravnost rada programa.

2. Definicija i opis zadatka

2.1. Uvod u problematiku

Metoda aukcije i slične metode namijenjene su sustavima s velikim brojem robota i poslova koje treba izvršiti te su zamišljene da budu decentralizirane. Nakon niza testiranja sustava koji funkcioniraju na sličan način, ispostavilo se da je decentralizacija najpraktičnije rješenje u sustavima s velikim brojem robota i zadataka. Decentralizacija također pruža jednostavnost pri praćenju cjelokupnog procesa u velikim sustavima isto kao i smanjen rizik od pogreške pri podjeli zadataka u odnosu na metode koje su izvedene na centraliziran način.

Osnovna je svrha ovakvih metoda, pa tako i metode aukcije, sve zadatke i poslove koje treba izvršiti ravnomjerno rasporediti robotima tj. agentima koji su u mogućnosti obaviti zadane poslove, pritom uzimajući u obzir vremenska ograničenja, sposobnosti svakog od robota, lokacije na kojima se nalaze roboti odnosno zadatci, prednosti u izvršavanju pojedinih zadataka itd. Metoda aukcije funkcionira na principu slanja ponuda za određene zadatke od strane robota koje su potom prihvaćene odnosno odbijene.

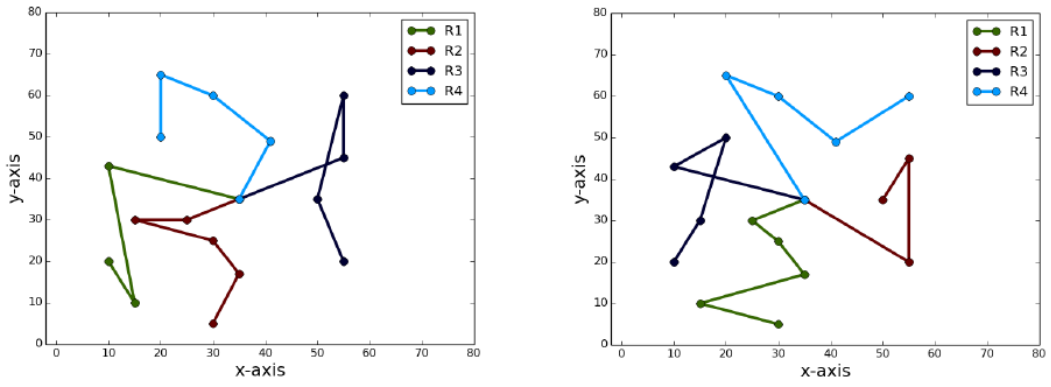
2.2. Opis zadatka

Osnovni cilj koji mora biti riješen ovom metodom sastoji se od n zadataka i m robota. Svaki od n zadataka ima određene parametre koji ga karakteriziraju i predstavljaju zahtjeve i uvjete pod kojima ih robot može riješiti. Dodatno imamo skup $\mathscr{R}$, skup od m robota od čega svaki ima svoju početnu lokaciju; početne lokacije se naravno mogu razlikovati, no bez obzira na to $\mathscr{R}$ je homogeni skup [4]. Također svaki se zadatak nalazi na određenoj lokaciji na kojoj se robot mora nalaziti da bi bio u mogućnosti riješiti taj zadatak [4].
Svaki član skupa $\mathscr{R}$ odnosno svaki robot posjeduje određene osobine koje ga karkteriziraju, to su redom: naziv koji je specifičan za svakog robota, brzinu kretanja

izraženu u metrima po sekundi, tip resursa kojeg robot koristi za ostvarivanje konekcije sa zadatkom te već spomenutu početnu lokaciju izraženu pomoću X i Y koordinate. Svi ovi navedeni parametri biti će ključni prilikom podjele zadataka.

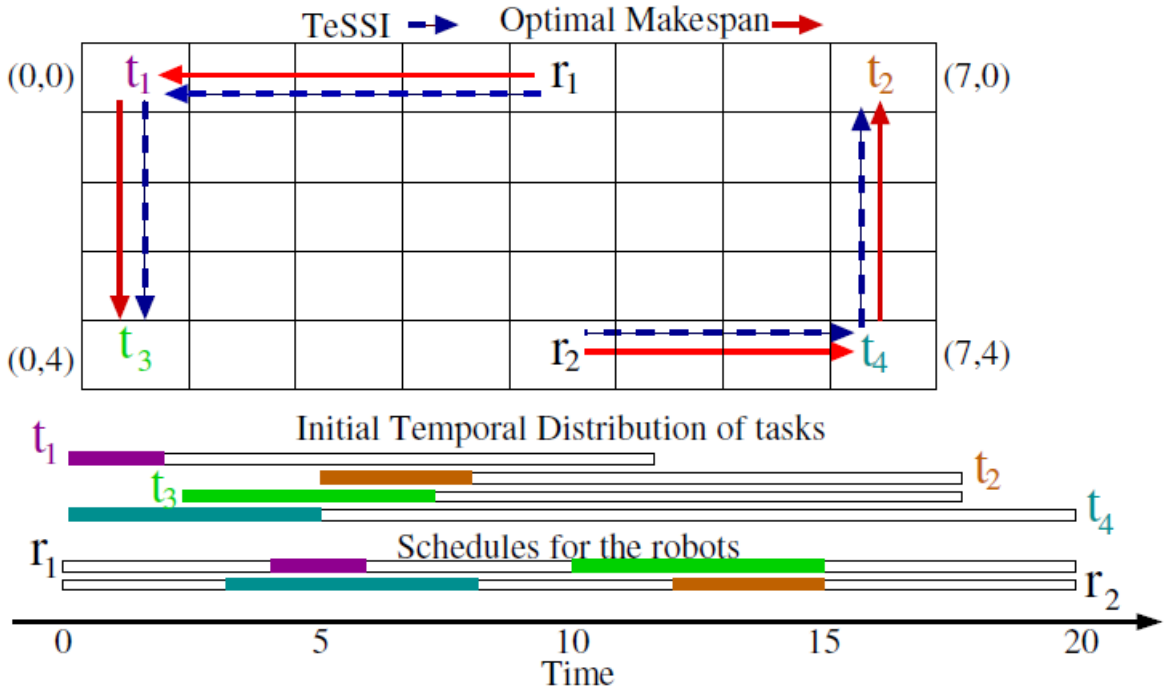
Svaki zadatak definiran je sljedećim parametrima: imenom koje je specifično za svaki pojedini zadatak, tipom resursa kojim se ostvaruje konekcija sa pojedinim robotom, potrebnim vremenom da se zadatak izvrši, zadatcima koji prethode tom zadatku, lokacijom zadatka te najranijim i najkasnijim trenutkom u vremenu u kojem zadatak mora biti izvršen.

Zadatak ovog projekta temelji se na prioritetima tj. sustav mora biti u stanju dostupne zadatke podijeliti na način da njihov redoslijed rješavanja bude optimalan. Svaki robot bi na kraju trebao imati otprilike jednako vremensko trajanje rasporeda kao i ostali roboti te bi svaki robot trebao "otići na svoju stranu" i rješavati zadatke koji su jedni drugima blizu kao što je prikazano na slici 2.1, no valja napomenuti da korisnik sustava zadaje uvjete i zahtjeve zadataka pa su stoga moguće različite situacije i rješenja u kojima će optimalan rezultat biti onaj s većom vremenskom razlikom.



Slika 2.1: Primjer optimalne rute robota (lijevi graf) i rute dobivene algoritmom iterirane metode aukcije s prioritetima (desni graf) [4]

Još jedna jako bitna stavka koja se mora uzeti u obzir u ovom procesu jesu vremenska ograničenja zadataka tj. vremenski prozori. Vremenski prozori predstavljaju period vremena u kojem zadatak mora biti riješen. Roboti odnosno agenti, uz sve ostale zahtjeve zadatka, moraju uzeti u obzir i vremenske prozore te vidjeti jesu li u stanju obaviti zadatak u zadanom vremenu. Primjer rasporeda s vremenskim prozorima prikazan je na slici 2.2.



Slika 2.2: Primjer rasporeda s vremenskim prozorima [7]

Zbog jednostavnosti, ali i zbog nedostatka vremena, nije bilo moguće testirati metodu na stvarnim robotima i zadatcima te zbog prevelikih zahtjeva koje bi postavio sustav s velikim brojem robota i zadataka. U dogovoru s mentorom, ovaj rad obrađuje pojednostavljeni model metode aukcije koji se testira na manjem broju robota i zadataka neumanjujući funkciju i samu svrhu metode aukcije te vjerodostojnost rezultata.

3. Metoda aukcije

3.1. Kratki opis metode

Metoda aukcije, kao što i samo ime kaže, bazira se na principu ponuda koje pristižu dok god ima nedodijeljenih zadataka koje su onda, ovisno o različitim parametrima i uvjetima, prihvaćene ili odbijene. Ovaj rad obrađuje centralizirani oblik metode aukcije s manjim brojem agenata koji šalju ponude i jednim aukcionarom koji nudi zadatke te prihvaća odnosno odbija pristigle ponude. Ovakav centralizirani oblik podrazumijeva da je lako implementirati da svaki agent može poprimiti ulogu aukcionara pa se ne gube svojstva decentraliziranosti.

Proces započinje na način da aukcionar rasporedi sve dostupne zadatke u tri skupine; prvu skupinu čine zadatci najvišeg prioriteta; to su zadatci koji nemaju svog prethodnika. Drugu skupinu čine zadatci koji svoje prethodnike imaju u prvoj skupini, te potom imamo ostale zadatke koji čine treću skupinu zadataka; zadatci u toj skupini svoje prethodnike imaju u prvoj i drugoj skupini. Nakon što su zadatci raspoređeni po skupinama, aukcionar započinje slati zadatke na aukciju određenim redoslijedom; taj redoslijed detaljnije je opisan u potpoglavlju 3.4.

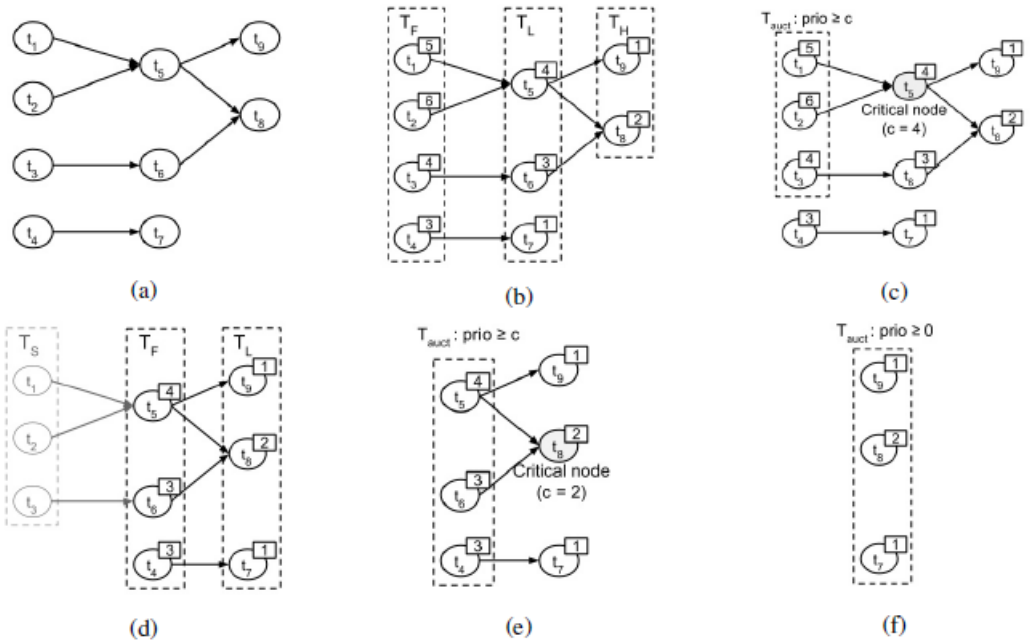
Nakon što su zadatci poslani na aukciju, roboti računaju ponude te ih potom šalju aukcionaru. Način na koji aukcionar odabire ponude vrlo je jednostavan; ukoliko je za određeni zadatak stigla samo jedna ponuda ona je automatski prihvaćena, a ako je za određeni zadatak stiglo više ponuda, aukcionar će prihvatiti ponudu s najkraćim vremenskim trajanjem. Proces se ponavlja dok god svi zadatci ne budu dodijeljeni. Na kraju procesa roboti izvještavaju kojim redoslijedom će rješavati zadatke u svojim rasporedima i koliko će raspored vremenski potrajati.

Detaljnija objašnjenja svakog pojedinog koraka u procesu, kao na primjer računanje i slanje ponude, određivanje prioriteta, računaje vremena, određivanje redoslijeda aukcije i tako dalje, nalaze se u sljedećim poglavljima.

3.2. Iterirana metoda aukcije

Naš algoritam temelji se na metodi iterirane aukcije, koja konstantno koristi modificiranu verziju SSI aukcije (engl. *Sequential Single Item*) kako bi rasporedila zadatke po skupinama. Svaki zadatak u određenoj skupini neovisan je o drugim zadatcima iz svoje skupine. To je glavni princip rada SSI aukcije [4]. Grafički prikaz metode pokazan je na slici 3.1.

Nakon što aukcionar ima informacije o svim zadatcima i poznaje njihove prethodnike, raspoređuje ih u tri skupine; zadatci koji nemaju prethodnika su tzv. "slobodni" zadatci koje algoritam stavlja u skupinu TF - to su zadatci najvišeg prioriteta koji moraju biti prvi riješeni jer o njima ovise ostali zadatci. Sljedeću skupinu - TL čine zadatci koji svoje prethodnike imaju u skupini TF. Svaki zadatak iz skupine TL može imati jednog ili više prethodnika. I na kraju, posljednju skupinu čine tzv. "skriveni" zadatci; to su zadatci koji svoje prethodnike imaju u skupini TL i TF te ih algoritam stavlja u skupinu zadataka najnižeg prioriteta - TH. Zadatci u skupini TH rješavaju se posljednji.



Slika 3.1: Grafički prikaz metode aukcije [4]

3.2.1. Jednostavni algoritam iterirane metode aukcije

Jednostavni algoritam iterirane metode aukcije (engl. *Simple Iterated Auction Algorithm - sIA*) predstavlja pojednostavljeni oblik metode koja se obrađuje u ovom radu, no za bolje razumijevanje i shvaćanje načina rada metode iterirane aukcije s prioritetima, potrebno je spomenuti i ovaj oblik metode. Osnova razlika jednostavnog algoritma i algoritma s prioritetima jest u tome što kod jednostavnog algoritma nemamo prioritete među zadatcima tj. svi zadatci imaju "nulti" prioritet [4]. Također, svi zadatci svrstani su u jednu skupinu te se svi istovremeno šalju na aukciju dostupnim robotima. Roboti potom računaju ponude prilikom čega ne uzimaju u obzir nikakve prethodnike niti skupinu kojoj zadatci pripadaju.

Ovakav oblik metode odličan je primjer za provjeru rada dosadašnjeg uratka za osobu koja radi na algoritmu s prioritetima jer se mogu provjeriti neke tehničke stvari koje su iste u oba algoritma kao npr. komunikacija između agenata i aukcionara; slanje zadataka na aukciju, slanje, prihvaćanje i odbijanje ponuda itd.

3.2.2. Algoritam iterirane metode aukcije s prioritetima

Algoritam iterirane metode aukcije s prioritetima (engl. *Iterated Auction with Prioritization algorithm - pIA*) osnovna je metoda koja se obrađuje u ovome radu i projektu. Algoritam pokušava ponuditi optimalno rješenje podjele zadataka uzevši u obzir prioritete prilikom slanja zadataka na aukciju. Redoslijed zadataka kojim će biti poslani na aukciju može se računati na više načina. U ovome radu kriterij po kojem se određuje redoslijed određen je veličinom grafa; jednostavnije rečeno - zadatci koji su dio grafa koji se proteže kroz sve tri skupine zadataka prvi će biti stavljeni na aukciju počevši od skupine TF [7]. Detaljniji prikaz moguće je vidjeti u poglavlju Rezultati i primjeri.

Zadatak aukcionara jest, nakon što je razvrstao zadatke po skupinama i odredio redoslijed slanja zadataka na aukciju, poslati zadatke zadanim redoslijedom i čekati ponude. Za to vrijeme roboti računaju i šalju svoje ponude nazad aukcionaru koji potom odabire pobjednike. Proces se ponavlja dok god svi zadatci nisu dodijeljeni. Ukoliko se dogodi situacija da određeni zadatak ne može biti dodijeljen, njega aukcionar uklanja iz aukcije i javlja korisniku da je došlo do pogreške prilikom zadavanja zahtjeva zadatka. Taj slučaj najčešće će se događati kada su vremenski prozori krivo zadani, no moguće je i da zbog nedovoljnog broja robota nije moguće ostvariti konekciju sa zadatkom.

Nakon što je aukcionar odredio pobjednike, potrebno je javiti robotima da su

njihove ponude izabrane odnosno odbijene. Svaki robot čija je ponuda prihvaćena mora raspored, zajedno sa zadatkom koji je prihvaćen, učiniti nepromjenjivim [4].

3.3. Računanje ponuda i slaganje rasporeda

Računanje ponuda i slanje konačne ponude jedan je od najvažnijih dijelova metode aukcije. U obliku metode koji se obrađuje u ovom radu roboti jednostavno dobiju popis zadataka od aukcionara, izračunaju vremenski najkraće trajanje trenutnog rasporeda sa svakim zadatkom koji je na aukciji i nakon toga pošalju ponudu. Ako je ponuda prihvaćena, robot stavlja pobjednički zadatak u svoj raspored [4].

Jedan od problema ovog algoritma jest u tome što roboti nisu svjesni prioriteta među zadatcima jer računaju isključivo najkraće trajanje rasporeda za svaki zadatak i potom zadatak s najkraćim vremenskim trajanjem rasporeda šalju kao ponudu [4]. Osim toga, zadatci također imaju i druga ograničenja koja, u određenim situacijama, sprječavaju robote da pošalju ponude za te zadatke. To je jedan od razloga zašto moramo uvesti određene provjere u algoritam za računanje najkraće ponude.

Osim računanja vremena, algoritam izvršava provjere valjanosti ponude tj. provjera ispunjava li robot zahtjeve zadatka. Ti zahtjevi mogu biti npr. tip konekcije kojom se mora ostvariti veza sa zadatkom da bi mogao biti riješen, provjera vremenskih prozora, odnosno hoće li robot biti u stanju riješiti zadatak u zadanom vremenu itd [3]. Ukoliko robot ne ispunjava sve uvjete zadatka, ne može poslati ponudu za taj zadatak. Ovakav oblik SSI algoritma naziva se ModifiedSSI algoritam [4].

Nakon što je ponuda prihvaćena, robot pomoću algoritma TightenSchedule zadatak stavlja u svoj raspored i taj raspored postaje njegov stvarni raspored kojeg više nije moguće promijeniti [4].

3.4. Određivanje prioriteta

Određivanje prioriteta je, uz već spomenut i objašnjen način računanja ponuda, najvažniji dio metode aukcije. Osnovni smisao ovog zadatka odnosno metode jest dodjeliti zadatke robotima uzevši u obzir prioritete među zadatcima te redoslijed kojim će biti stavljani na aukciju [8]. Moglo bi se reći i da kvaliteta ove metode leži u načinu određivanja prioriteta. Stoga je važno dobro odraditi ovaj dio metode

prilikom izrade sustava koji se temelji na metodi aukcije s prioritetima. Kao što je već ranije rečeno, osnovni kriterij korišten u ovome radu, po kojem se određuje redoslijed zadataka kojim će biti stavljani na aukciju jest veličina grafa.

Zadatci se moraju stavljati na aukciju pravilnim redoslijedom, to znači da se moraju poštivati prioritetne skupine; TF, TL i TH. No uz to algoritam gleda i veličinu grafa kojem pripada pojedini zadatak i na temelju toga slaže konačne grupe zadataka za aukciju. Ukoliko je zadatak dio grafa koji se proteže kroz sve tri skupine, tada se zadatci iz tog grafa redom šalju na aukciju poštujući prioritetne skupine. Ako se graf proteže kroz dvije skupine, tada se zadatci iz tog grafa smatraju zadatcima drugog prioriteta te oni idu na aukciju nakon zadataka prvog prioriteta također takvim redoslijedom da se poštuju prioritetne skupine. Zadatci koji ne ovise niti o jednom drugom zadatku niti o njima ovisi neki drugi zadatak, su zadatci najnižeg prioriteta te na aukciju idu posljednji. Sve rečeno u ovome odlomku grafički je prikazano u poglavlju Rezultati i primjeri.

Još je jednu stavku bitno naglasiti prilikom objašnjavanja i određivanja prioriteta, a to je odgovornost korisnika sustava. Osoba koja koristi sustav koji funkcionira na ovakav način, tj. sustav koji se temelji na metodi aukcije, mora biti odgovorna prilikom određivanja zahtjeva zadataka. Sustav je napravljen na način na koji je napravljen i ne može raditi nikako drugačije što znači da su moguće situacije u kojima korisnik neće dobro definirati zadatke i doći će do situacije u kojoj niti jedan robot neće moći ispuniti sve uvjete da bi mogao poslati ponudu za određeni zadatak što ostavlja taj zadatak nerješivim. U takvoj situaciji proces se nastavlja normalno odvijati, ali aukcionar takve nerješive zadatke uklanja iz popisa zadataka za aukciju i javlja korisniku da ti zadatci ne mogu biti riješeni.

4. Implementacija

4.1. Podatci

Za potrebe ovog završnog projekta, svi podatci o dostupnim zadatcima i robotima zapisani su u datoteci Tasks.yaml. Sve su karakteristike robota kao i zadataka pretpostavljene za potrebe ovog projekta, no u stvarnom slučaju odnosno kada bi program testirali na pravim robotima i zadatcima, te bi karakteristike bile gotovo pa identične te stoga podatci dostupni u našoj datoteci ne umanjuju vjerodostojnost rezultata.

Tasks.yaml datoteka podijeljena je na dva dijela; prvi dio pod imenom ”agents” sastoji se od dostupnih robota i njihovih karakteristika dok drugi dio ”tasks” čine zadatci koje treba dodijeliti robotima. Svaki robot predstavljen je karakteristikama ”name”, ”velocity”, ”resources” i ”location” koje predstavljaju njegovo ime, brzinu kretanja, tip konekcije koju može ostvariti i početnu lokaciju. Svaki zadatak sadrži karakteristike ”name”, ”resources”, ”duration”, ”precedence”, ”earliest_start_time”, ”latest_finish_time” i ”location” gdje ”name”, ”resources” i ”location” predstavljaju isto što i kod robota, a ostale karakteristike predstavljaju vrijeme trajanja zadatka, njegove prethodnike te vremenski prozor u kojem zadatak mora biti riješen.

Prije početka izvođenja drugih funkcija i operacija, aukcionar mora učitati navedenu datoteku da bi dobio informacije o zadatcima. Također, svaki *node* tj. proces koji izvodi određenu radnju, stvara vlastiti objekt tipa Robot pomoću karakteristika robota navedenih u Tasks.yaml datoteci.

4.2. Programska izvedba

Programska izvedba ovog završnog projekta strukturirana je na sljedeći način:
config (konfiguracijske datoteke s opisom zadataka i robota)

- Tasks.yaml

launch (datoteke koje pokreću skripte)

- pia_ros.launch

msg (definicije ROS poruka)

- TaskMessage.msg

scripts (glavne programske datoteke)

- Auctioneer_node.py
- Bid.py
- Bidder_node.py
- Task.py

srv (definicije ROS servisa)

- DeclareEnd.srv
- DeclareWinner.srv
- ProvideTauc.srv
- TightenSchedule.srv

Program je u cijelosti napisan u programskom jeziku Python. Program koristi biblioteke za računanje matematičkih funkcija i matplotlib [6] za grafički prikaz rezultata. Također program ne može raditi bez uključenog Robotskog Operacijskog Sustava (ROS) jer se za prijenos parametara na server, isto kao i za komunikaciju između robota i aukcionara, koriste metode i funkcije iz biblioteke rospy [1] te ROS Servisi [2].

Za pravilno pokretanje programa potrebno je imati ROS paket naziva pia_ros unutar radnog prostora u kojem će biti smješteni gore navedene mape s odgovarajućim datotekama. Također potrebno je prilagoditi datoteke CMakeLists.txt i package.xml. Cijeli navedeni paket koji je u potpunosti funkcionalan dostupan je za preuzimanje na poveznici `https://github.com/FilipSkoro/pia_ros`.

4.3. Tok programa

Program se pokreće pokretanjem pia_ros.launch datoteke koja potom izvodi nekoliko naredbi. Prvo učitava podatke iz datoteke Tasks.yaml na server odakle će *node* tj. proces koji izvodi određenu radnju, temeljen na skripti Auctioneer_node.py, nakon što stvori objekt klase Auctioneer, učitati podatke o zadatcima i svaki zadatak pretvoriti u objekt klase Task. Također *launch* datoteka pokreće i onoliko instanci Bidder_node.py koliko imamo dostupnih robota. Svaki *node* koji se temelji na datoteci Bidder_node.py stvara po jedan objekt klase Robot i takav *node* onda predstavlja jednog robota. Podatci o robotima se također uzimaju sa servera. Nakon što je aukcionar učitao i pripremio zadatke može se započeti s procesom aukcije pod ranije opisanim pravilima.

4.3.1. Programsko određivanje prioriteta i redoslijed aukcije

Kao što je već prije spomenuto, aukcionar mora rasporediti zadatke i odrediti redoslijed aukcije. U programskom smislu, nakon što je učitao sve zadatke, aukcionar koristi dvije funkcije. Prvo poziva funkciju sort_Tasks koja zadatke razvrstava u liste TF, TL i TH. Potom poziva funkciju define_Tauc koja određuje redoslijed slanja zadataka na aukciju po principu kako je objašnjeno ranije. U programu to se odvija na način da aukcionar modificira liste TF, TL i TH te ih onda redom šalje na aukciju.

Nakon što je zaprimio ponude, aukcionar određuje pobjednike i vraća robotima povratnu informaciju. Prilikom svakog koraka aukcije aukcionar obavještava korisnika programa od trenutnom stanju zadataka na aukciji dok roboti obavještavaju o trenutnom stanju svojih rasporeda kao i o tome je li njihova ponuda prihvaćena ili ne. Kada su svi zadatci iz određene liste dodijeljeni ili uklonjeni, ako niti jedan robot nije mogao poslati ponudu za njih, aukcionar stavlja na aukciju iduću listu i to se ponavlja dok svi zadatcu nisu raspoređeni.

Na kraju programa, aukcionar pomoću servisa DeclareEnd obavještava sve robote da je proces došao do kraja. Roboti potom vraćaju povratnu informaciju aukcionaru u obliku svojih rasporeda koje aukcionar onda grafički iscrtava u koordinatnom sustavu te također ispisuju redoslijed rješavanja zadataka u svojim rasporedima isto kao i vrijeme trajanja rasporeda kako bi i korisnik programa mogao vidjeti konačni rezultat.

4.3.2. Komunikacija između node-ova

Komunikacija između aukcionara i robota odvija se pomoću ROS servisa. Osim već ranije spomenutog DeclareEnd servisa kojim aukcionar javlja robotima da je aukcija završila, koriste se još tri servisa.

Servis DeclareWinner koristi se u svakom koraku aukcije i služi robotima za dobivanje informacije o tome je li njihova ponuda prihvaćena ili ne. Već je ranije rečeno da roboti moraju "fiksirati" svoje rasporede nakon svake prihvaćene ponude kako više ne bi moglo doći do promjene. To je zadaća servisa TightenSchedule kojim aukcionar javlja robotima da je završena aukcija za određenu prioritetnu skupinu i da se sada prelazi na sljedeću. To je znak robotima da sljedeći zadatci za koje će slati ponude moraju biti stavljani u raspored nakon već postojećih zadataka u rasporedu. Posljednji je servis ProvideTauc koji služi za prijenos informacija od aukcionara do robota o tome koji su zadatci trenutno na aukciji. Taj servis koristi tip poruke TaskMessage koji služi za prijenos objekata klase Task.

4.3.3. Računanje i slanje ponuda

Roboti zaprimaju zadatke koji su na aukciji pomoću ProvideTauc servisa i potom računaju ponude. Prva stvar koju svaki robot napravi jest da za svaki zadatak provjeri može li poslati ponudu za njega. To se programski odvija pozivanjem funkcije is_Valid u kojoj robot provjerava tip konekcije koji se ostvaruje sa zadatkom i duljinu vremenskog prozora. Ako funkcija vrati rezultat False, robot ne može poslati ponudu za taj zadatak te prelazi na sljedeći. Ako vrati True, robot nastavlja računati ponudu na način da zadatak umetne na svako mjesto u privremenom rasporedu, koji predstavlja njegov stvarni raspored, te računa na kojoj poziciji će raspored trajati najkraće. Kada je izračunao najkraće trajanje, robot šalje svoju ponudu.

Aukcionar u sklopu funckije sIA_function kojom se pokreće cijeli proces poziva funkcije choose_Bid i remove_Task nakon što je zaprimio ponude pomoću kojih određuje pobjednika i uklanja zadatak s aukcije ako je prihvaćen. Tada se poziva servis DeclareWinner i proces se ponavlja do zadnjeg zadatka.

5. Rezultati i primjeri

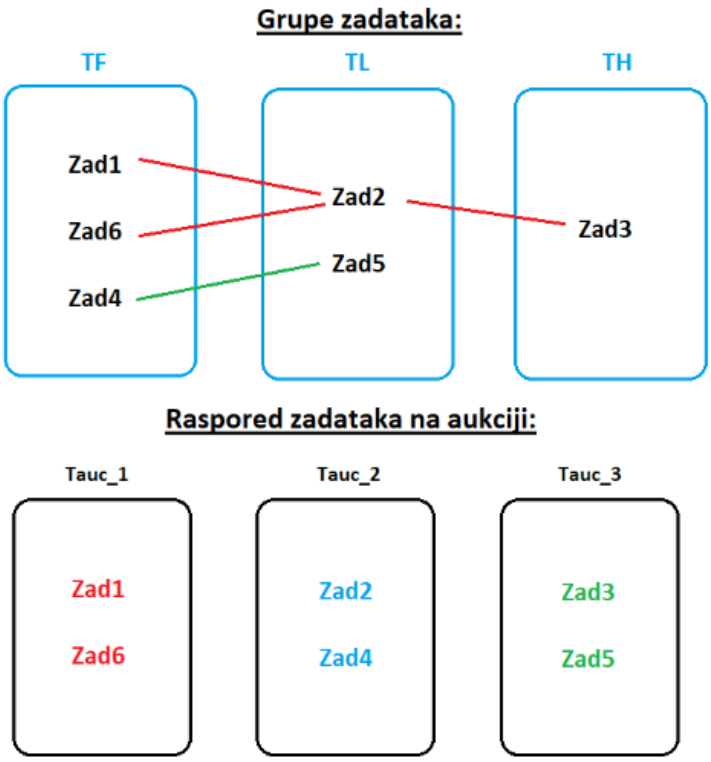
Prikazani primjeri u ovom poglavlju razlikuju se po ovisnostima među zadatcima te je prikazan jedan primjer s većim brojem robota i zadataka kako bismo potvrdili da program obavlja svoju funkciju i za velike sustave. Također primjeri se razlikuju i po vremenskim prozorima te u određenim situacija neće biti moguće riješiti sve zadatke.

Za svaki primjer prikazan je ručno napravljen grafički prikaz s jasnije izraženim elementima koji su nam bitni. Ručno izrađen prikaz također prikazuje i vremenske prozore za pojedini zadatak, ukupno trajanje rasporeda svakog robota, međuovisnosti zadataka te redoslijed slanja zadataka na aukciju.

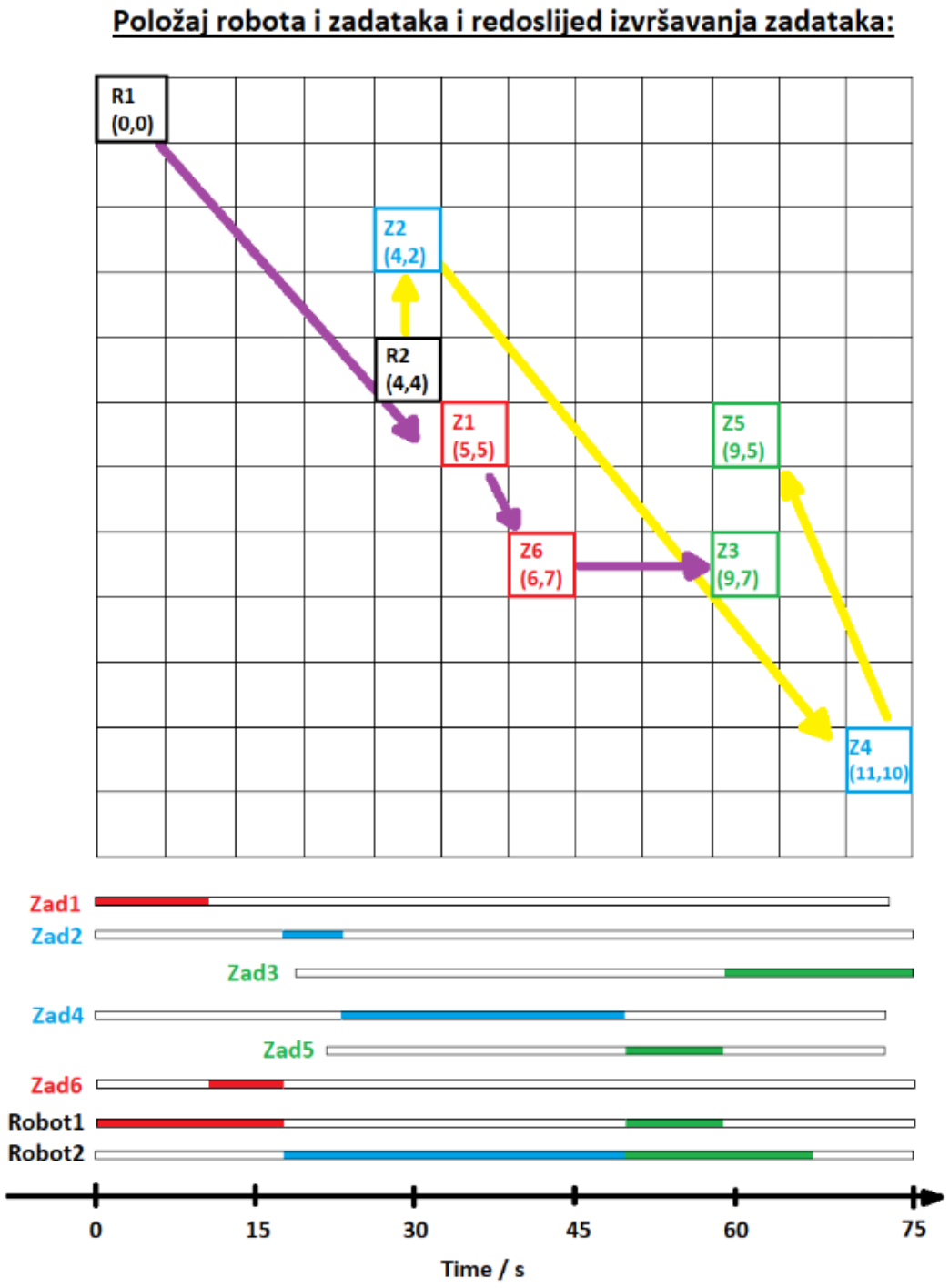
Valja napomenuti da neki roboti ni u kojem slučaju ne mogu slati ponude za određene zadatke pod uvjetima koji su zadani u sljedećim primjerima. Osim vremenskih prozora koji, ovisno o situaciji, mogu ili ne moraju predstavljati ograničenje, tip konekcije unaprijed je definiran i za robote i za zadatke. U sljedećim primjerima Robot1 jedini je agent koji može ostvariti konekciju koristeći wifi i bluetooth za razliku od ostalih robota koji su po načinu konekcije ograničeni samo na jedan tip. Stoga Robot1 jedini može slati ponude za svaki zadatak dok ostala tri robota mogu slati samo za zadatke koji imaju isti tip konekcije kao i oni. To može biti razlog većeg broja zadataka u rasporedu određenih robota u nekim primjerima.

5.1. Primjer 1

U ovom primjeru na slici 5.1 vidimo međiovisnosti zadataka te redoslijed kojim će biti stavljani na aukciju. Na slici 5.2 vidimo ručno izrađen prikaz koordinatnog sustava s vremenskim prozorima za svaki zadatak i raspored robota. Na grafu se vidi putanja kojom će se roboti kretati kao i njihov početni položaj te položaj zadataka. Ovaj primjer dobar je prikaz ranije objašnjenog rada algoritma koji određuje redoslijed slanja zadataka na aukciju. Kao i u ostalim primjerima, poanta je dokazati da svi algoritmi, kao i cijela metoda, rade ispravno.



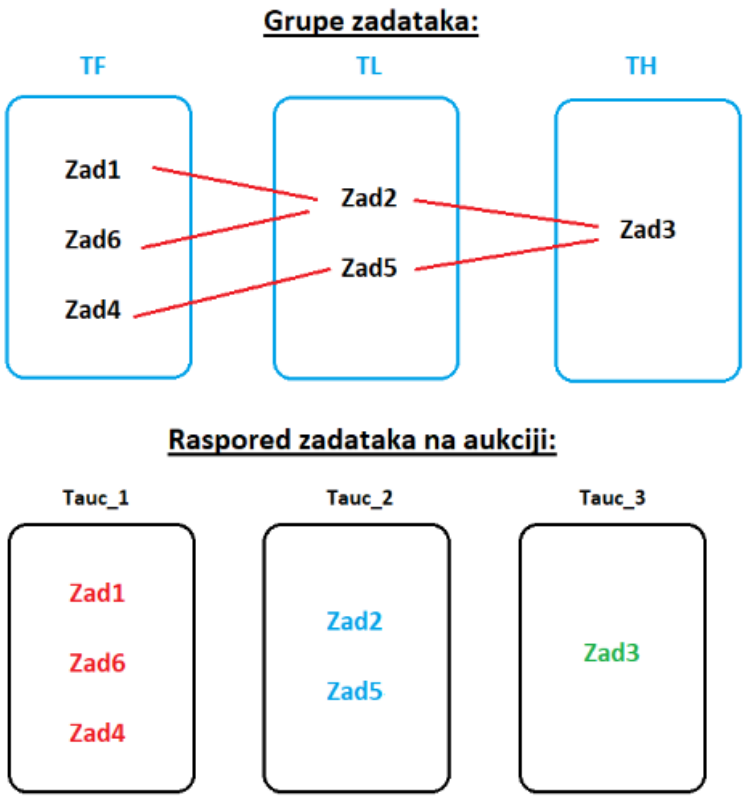
Slika 5.1: Međuovisnosti zadataka i raspored po skupinama - primjer 1



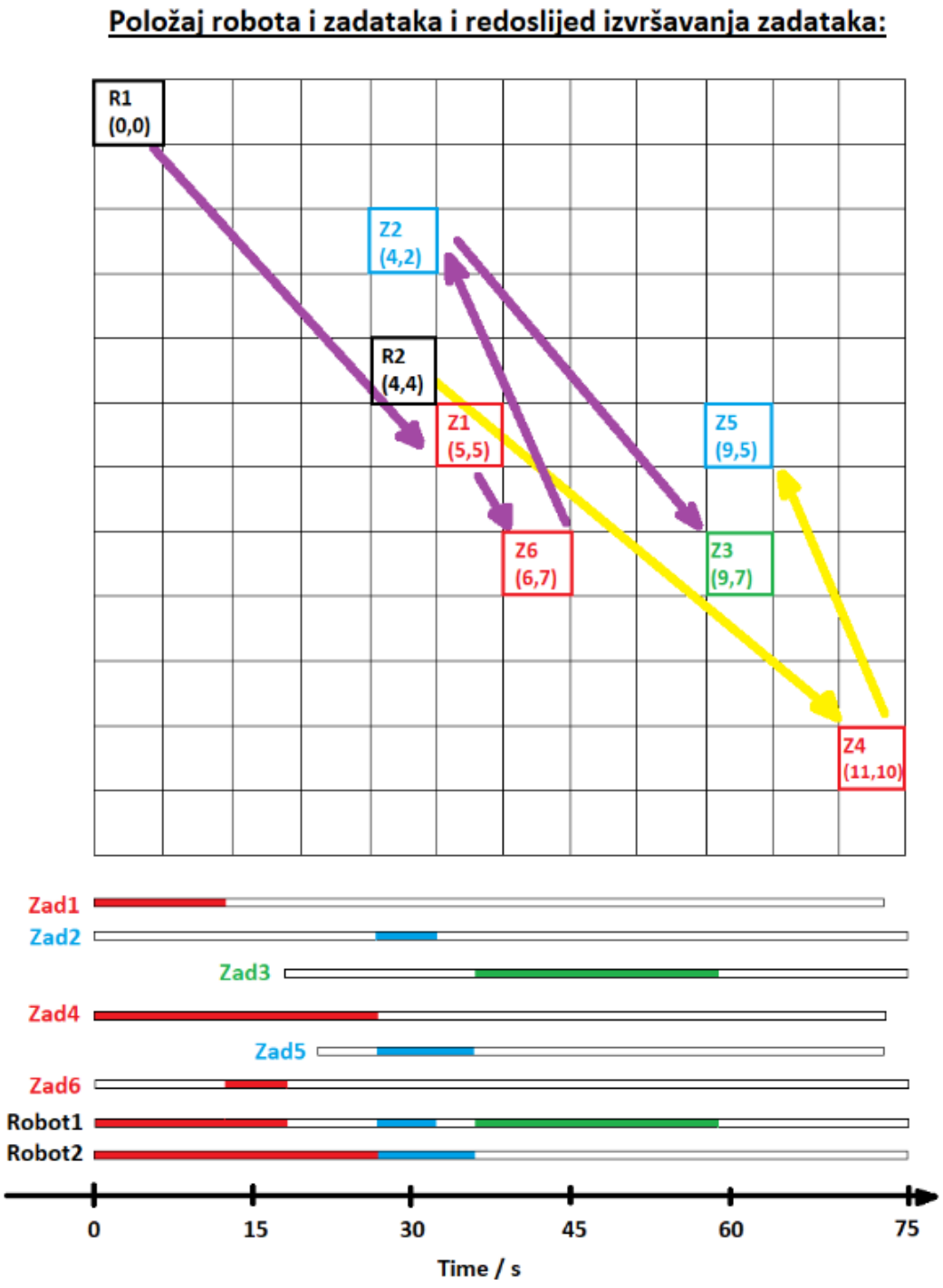
Slika 5.2: Ručno izrađen prikaz s vremenskim prozorima - primjer 1

5.2. Primjer 2

Sljedeći primjer prikazuje direktnu povezanost i ovisnost svih dostupnih zadataka. Na slici 5.3 vidimo da se graf proteže kroz sve tri prioritetne skupine. U takvom slučaju lako je zaključiti da će redoslijed zadataka na aukciji biti jednak onome kako su raspoređeni po prioritetnim skupinama. Sukladno tome grafički prikaz izgleda kako je priloženo na slici 5.4.



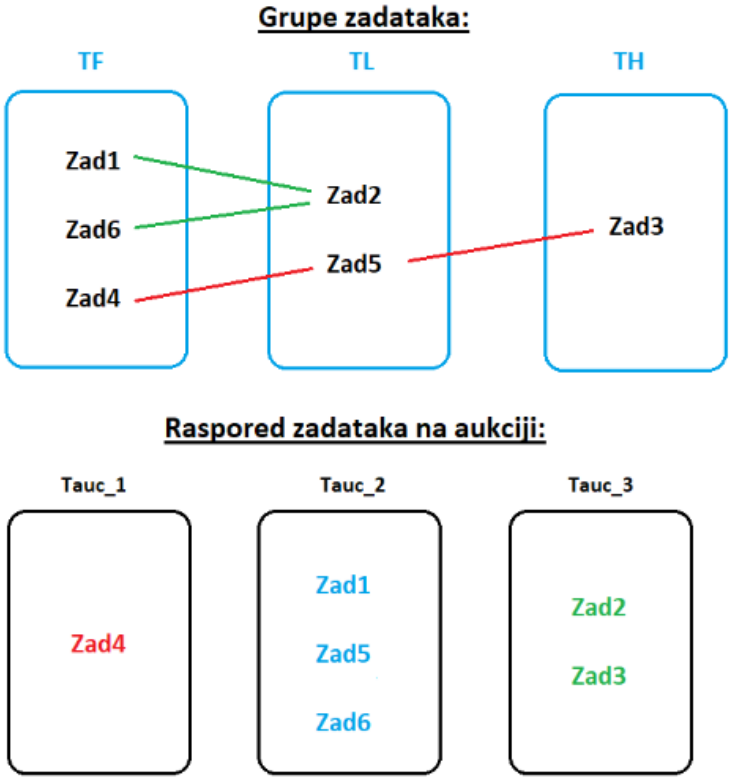
Slika 5.3: Međuovisnosti zadataka i raspored po skupinama - primjer 2



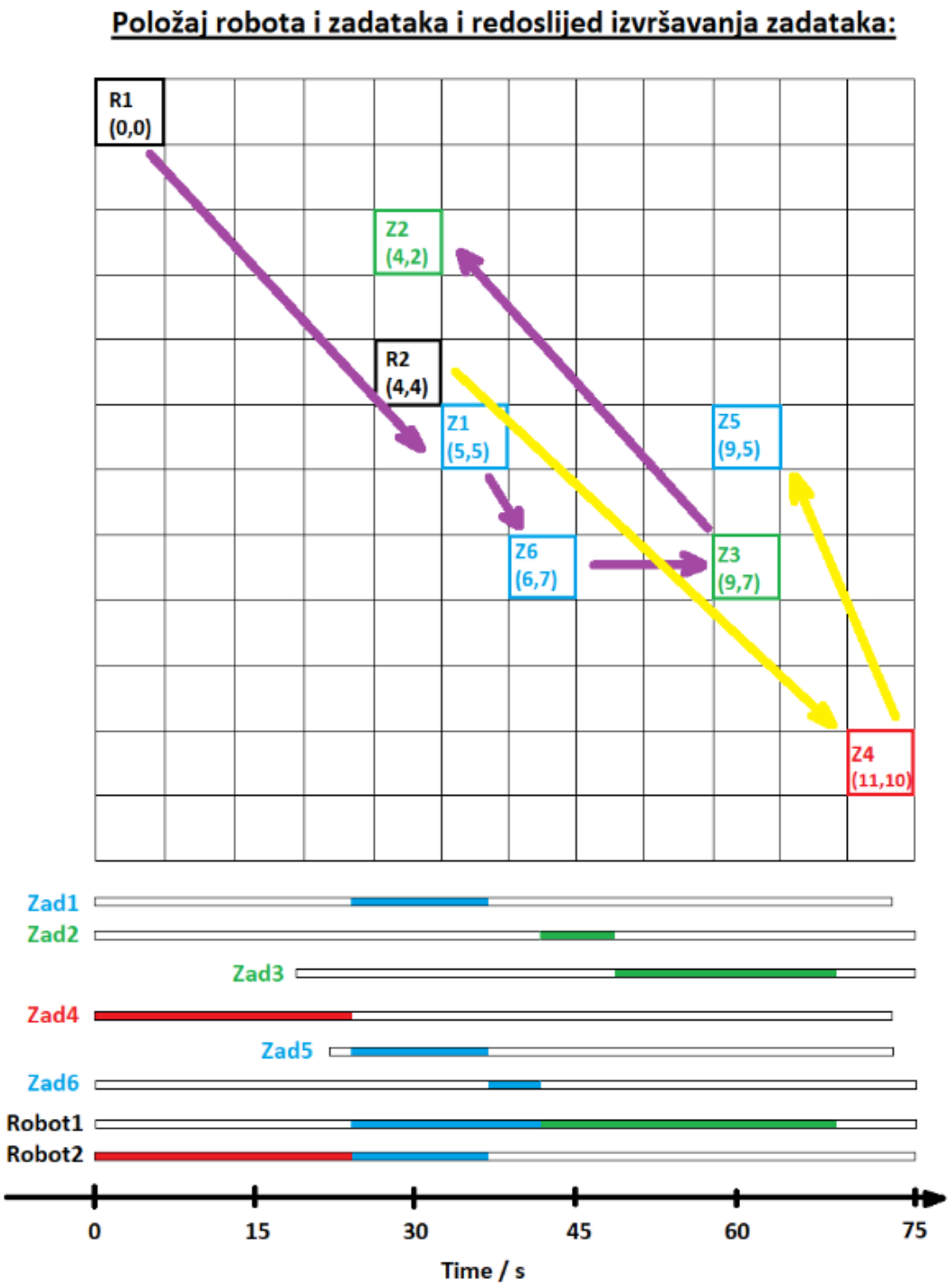
Slika 5.4: Ručno izrađen prikaz s vremenskim prozorima - primjer 2

5.3. Primjer 3

Ovaj primjer direktno se nadovezuje na prva dva primjera jer prikazuje minimalne promjene ovisnosti zadataka kako je prikazano na slici 5.5, no na grafičkom prikazu 5.6 vidi se da će redoslijed izvršavanja zadataka tj. putanje robota te vremenski prozori biti drugačiji u odnosu na prethodne primjere.



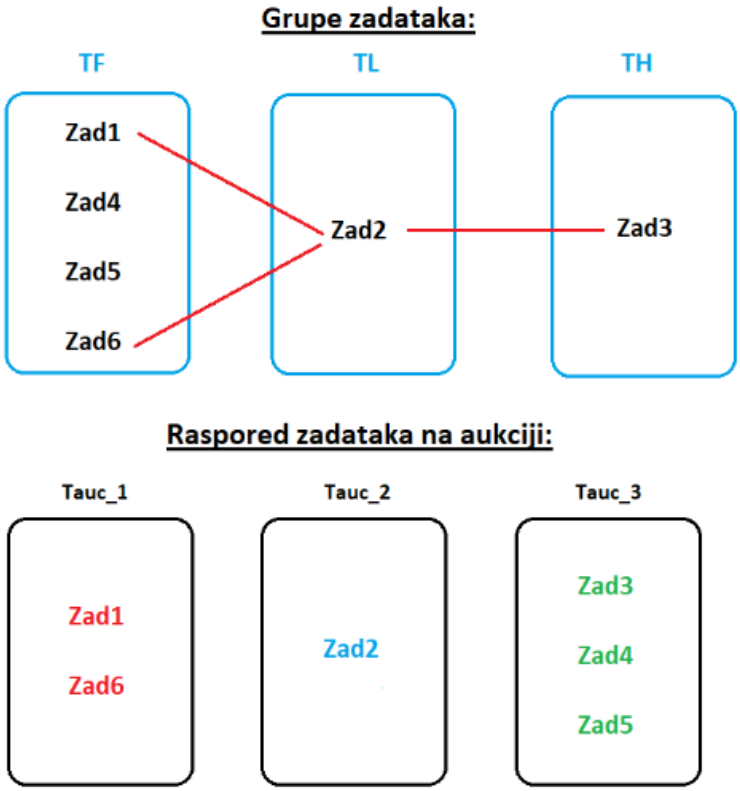
Slika 5.5: Međuovisnosti zadataka i raspored po skupinama - primjer 3



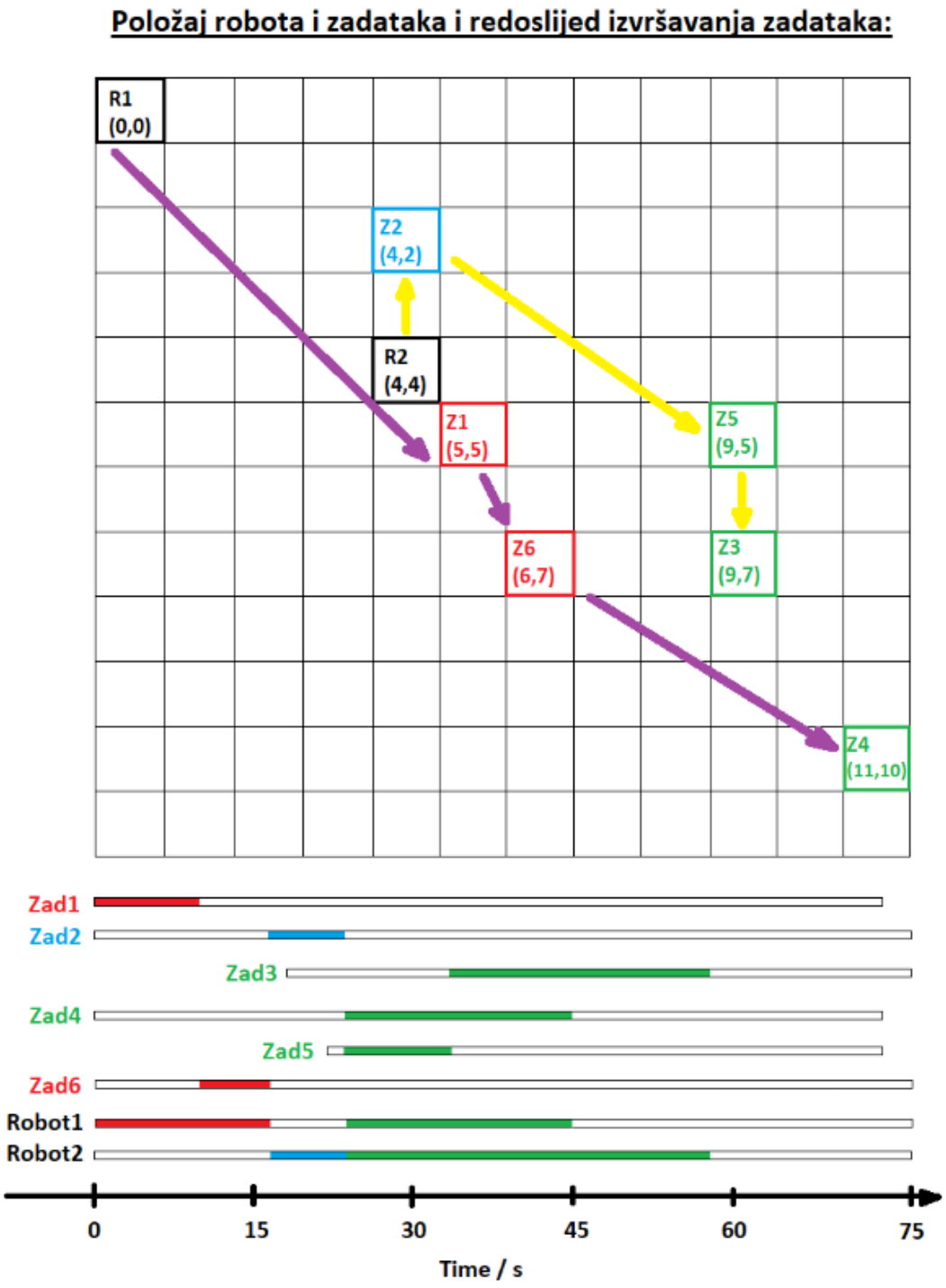
Slika 5.6: Ručno izrađen prikaz s vremenskim prozorima - primjer 3

5.4. Primjer 4

U ovom primjeru imamo situaciju u kojoj postoje čak dva zadataka koja su u potpunosti slobodna; ne ovise niti o jednom zadatku niti o njima ovisi neki zadatak. Sukladno tome, kako je ranije objašnjeno, dobivamo sljedeću raspodjelu zadataka po skupinama za aukciju što se vidi na slici 5.7. Grafičko rješenje s vremenskim prozorima za ovaj primjer prikazano je na slici 5.8.



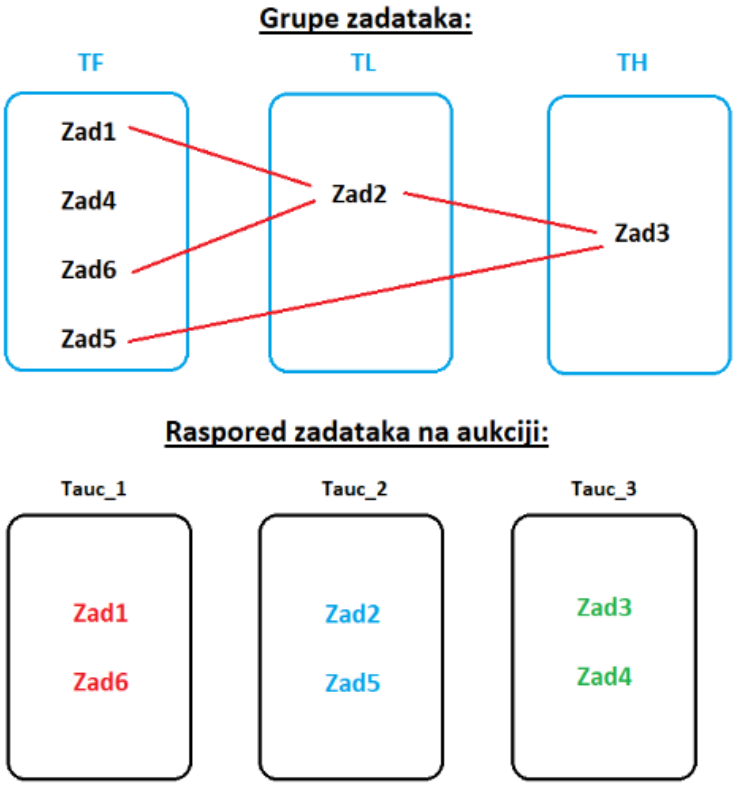
Slika 5.7: Međuovisnosti zadataka i raspored po skupinama - primjer 4



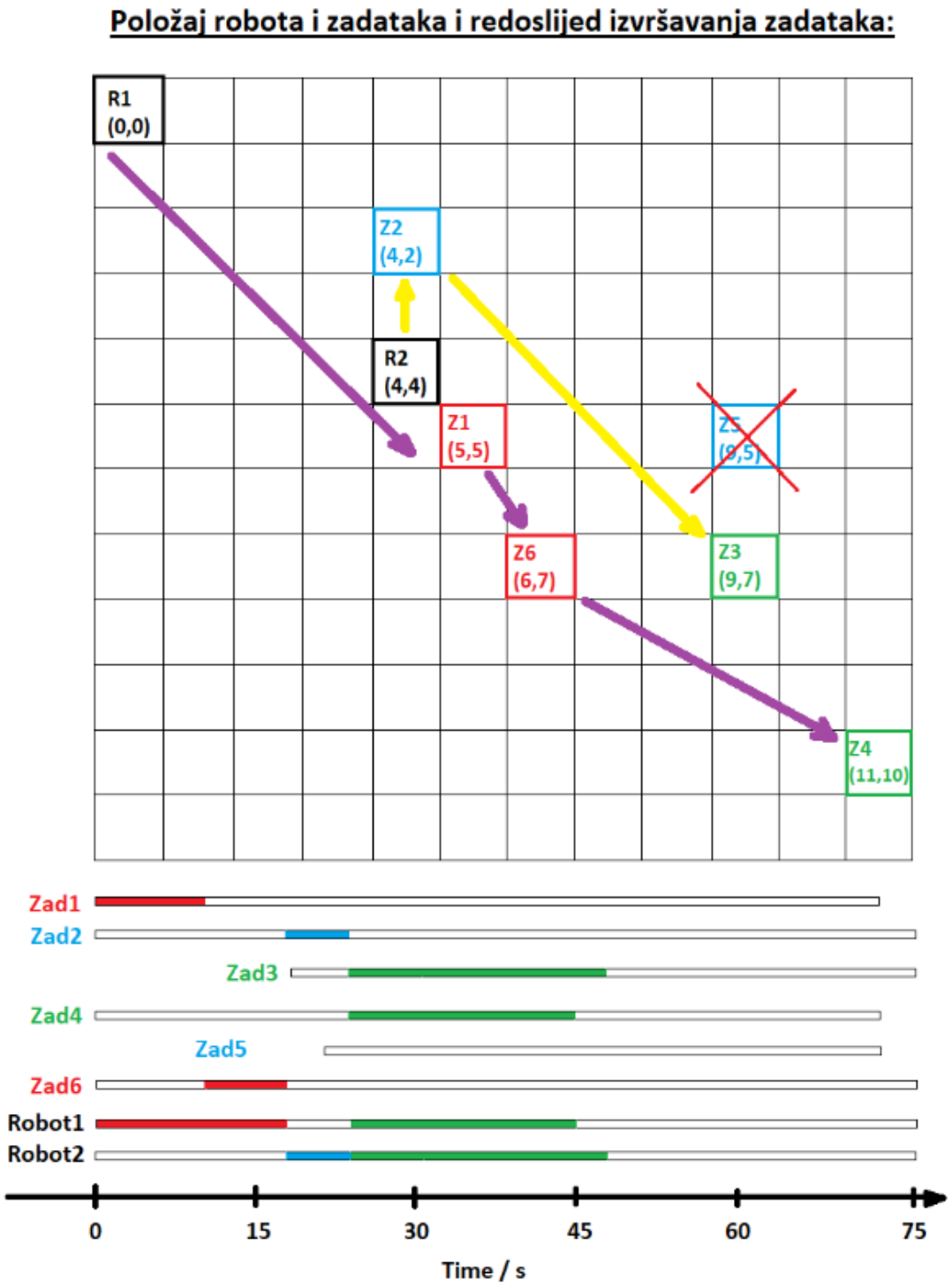
Slika 5.8: Ručno izrađen prikaz s vremenskim prozorima - primjer 4

5.5. Primjer 5

Sljedeći primjer je važan jer se prikazuje situacija o kojoj je bilo riječi u ranijim poglavljima, a to je slučaj u kojem imamo zadatak koji zbog nekih svojih zahtjeva, konkretno u ovom slučaju zbog vremenskog prozora i razine prioriteta, neće moći biti dodijeljen niti jednom robotu pa će ga stoga aukcionar izbaciti iz popisa zadataka za aukciju. U takvom slučaju aukcionar će korisnika programa obavijestiti da je došlo do pogreške odnosno nemogućnosti zadovoljavanja svih zahtjeva zadatka. Bez obzira na ovakvu situaciju, redoslijed slanja zadataka na aukciju odvija se po istom principu kao i u drugim primjerima 5.9 te je grafički prikaz predstavljen na slici 5.10.



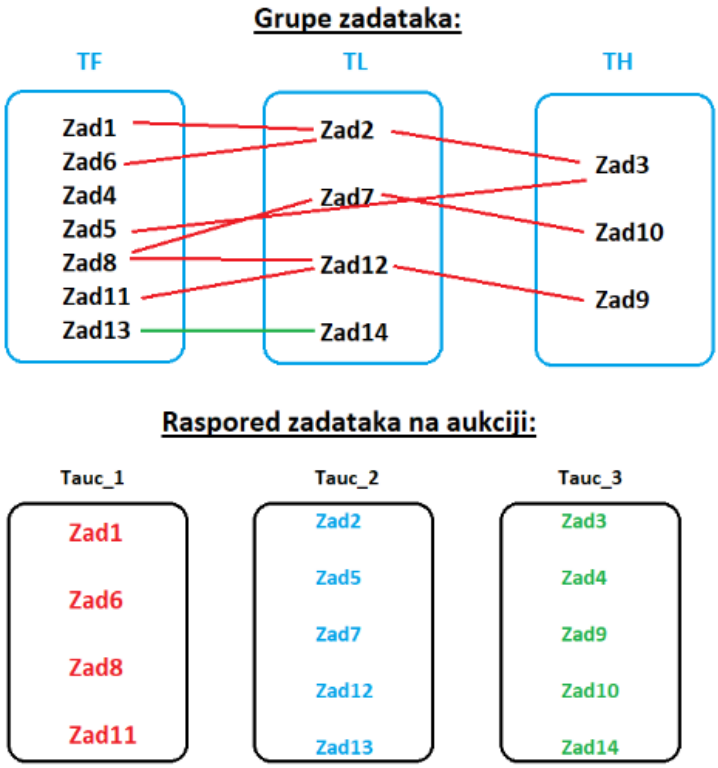
Slika 5.9: Međuovisnosti zadataka i raspored po skupinama - primjer 5



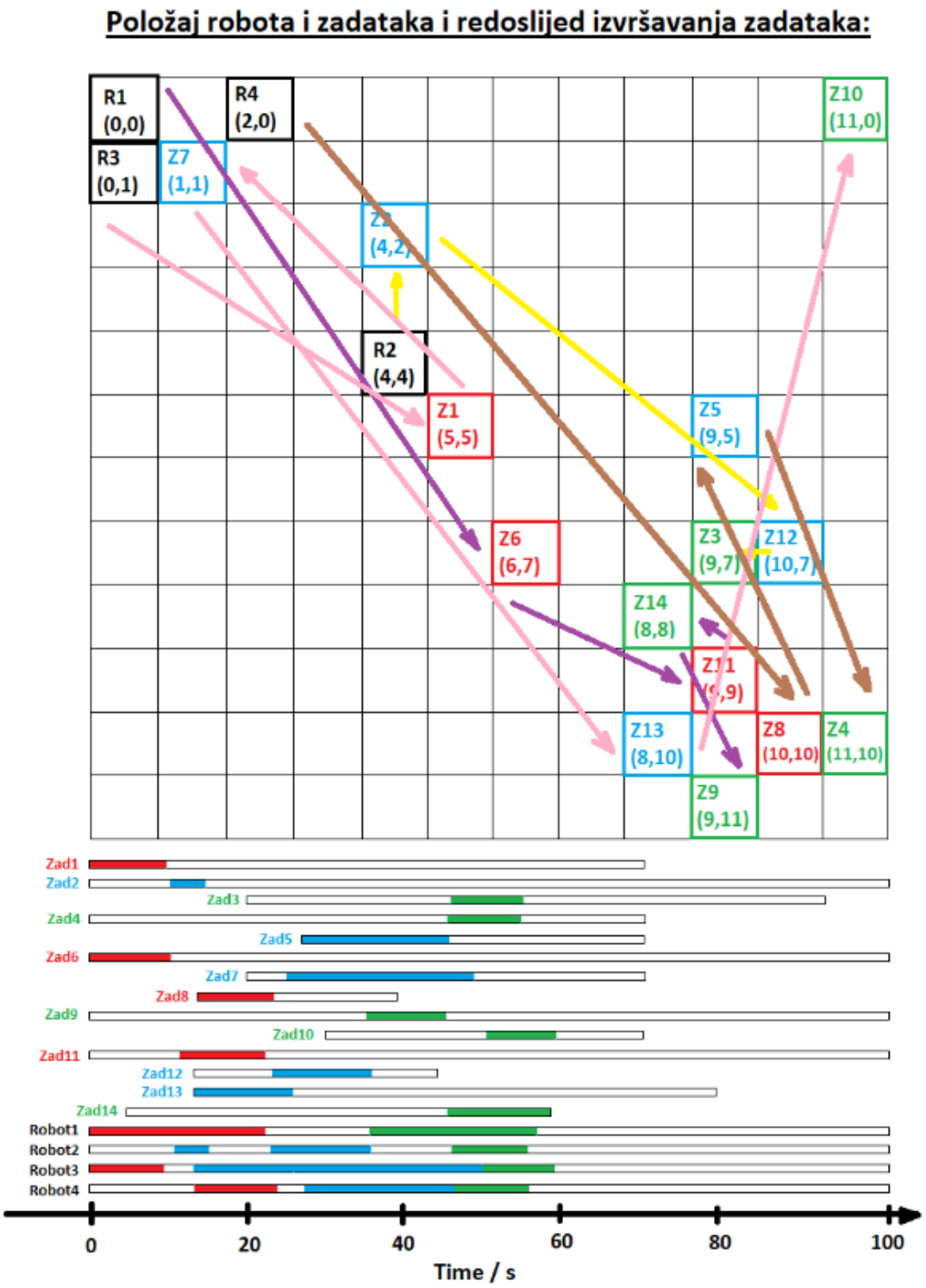
Slika 5.10: Ručno izrađen prikaz s vremenskim prozorima - primjer 5

5.6. Primjer 6

Ovaj primjer služi kao pokazatelj ispravnog rada metode na većem broju robota i zadataka. Na slici 5.11 prikazana je međuovisnost zadataka gdje se vide raznoliki odnosi među zadatcima što svakako predstavlja veći izazov za algoritam nego prethodne situacije. Također je prikazan i redoslijed slanja zadataka na aukciju. Na slici 5.12 vidi se graf koji prikazuje putanje, položaje robota i zadataka te vremenske prozore.



Slika 5.11: Međuovisnosti zadataka i raspored po skupinama - primjer 6



Slika 5.12: Ručno izrađen prikaz s vremenskim prozorima - primjer 6

6. Zaključak

Nakon objašnjenja svih potrebnih elemenata metode poput algoritama, funkcija, programske izvedbe, nakon objašnjenja potrebe i motivacije za ovakvom metodom te nakon prikazanih i objašnjenih rezultata, možemo zaključiti da je ovakav način izvedbe metode aukcije funkcionalan i primjenjiv na stvarne situacije. Metoda radi ispravno što je potvrđeno prikazanim primjerima.

Metoda aukcije ima određene prednosti i mane koje je moguće zaključiti iz poznavanja načina rada metode i rezultata. Prednosti metode aukcije svakako leže u tome što je ona po prirodi decentralizirana i svaki robot može slati ponude za svaki zadatak, naravno ako ga neka druga ograničenja ne sprječavaju u tome. Također jedna od prednosti je u tome što se zadatci dijele u prioritetne skupine što itekako olakšava situaciju i aukcionaru i robotima jer su zadatci unutar istih skupina međusobno nezavisni i mogu biti rješavani bilo kakvim redoslijedom.
S druge strane gledano, mane pronalazimo u tome što "roboti ne gledaju dovoljno u budućnost" što se tiče lokacije zadataka pa su moguće situacije u kojima će roboti zadatke izvršavati dužim putem jer su zbog prioriteta među zadatcima složili takav raspored [4].

Uzevši u obzir sve navedeno i prikazano, smatram da metoda aukcije pruža adekvatno rješenje stvaranja optimalnog rasporeda zadataka u više-robotskim sustavima te da je takvo rješenje jednako dobro kao i neka druga rješenja koja pružaju metode koje funkcioniraju po drugačijim principima. Ovakva programska izvedba metode aukcije otvorena je za daljnji napredak tj. moguće ju je dodatno unaprijediti dodavanjem novih algoritama i opcija kako bi bila sposobna napraviti optimalan raspored zadataka u još kompleksnijim sustavima.

Literatura

[1] rospy - ros wiki. URL `http://wiki.ros.org/rospy`. Datum pristupanja: 3.6.2021.

[2] services - ros wiki. URL `http://wiki.ros.org/Services`. Datum pristupanja: 3.6.2021.

[3] G. Ayorkor Korsah, M. Bernardine Dias, i Anthony Stentz. A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research*, 2013.

[4] Ernesto Nunes Mitchell McIntire i Maria Gini. Iterated Multi-Robot Auctions for Precedence-Constrained Task Scheduling. *AAMAS '16: Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, 2016.

[5] Seenu N., Kuppan Chetty R.M., Ramya M.M., i Mukund Nilakantan Janardhanan. Review on state-of-the-art dynamic task allocation strategies for multiple-robot systems. *Industrial Robot: the international journal of robotics research and application*, 47(6):929–942, Rujan 2020. doi: 10.1108/ir-04-2020-0073. URL `https://doi.org/10.1108/ir-04-2020-0073`.

[6] NumFOCUS. Matplotlib: Visualization with python. URL `https://matplotlib.org/`. Datum pristupanja: 3.6.2021.

[7] Ernesto Nunes i Maria Gini. Multi-robot auctions for allocation of tasks with temporal constraints. *AAAI'15: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[8] Ernesto Nunes, Mitchell McIntire, i Maria Gini. Decentralized allocation of tasks with temporal and precedence constraints to a team of robots. U *2016 IEEE International Conference on Simulation, Modeling, and Programming*

for Autonomous Robots (SIMPAR). IEEE, dec 2016. doi: 10.1109/simpar.2016.7862396. URL https://doi.org/10.1109/simpar.2016.7862396.

Raspoređivanje zadataka u više-robotskom sustavu korištenjem metode aukcije

Sažetak

Velika postrojenja poput tvornica, bolnica, skladišta nerijetko koriste uređaje odnosno robote koji automatski odrađuju poslove. Takva postrojenja često nailaze na probleme oko organizacije i podjele poslova u više-robotskim sustavima. Ovaj rad bavi se istraživanjem jedne od metoda koja pruža optimalan raspored zadataka - metoda aukcije. Objašnjena je problematika, potom teoretske postavke algoritama te programska izvedba. Rezultati razvijene metode predstavljeni su na 5 različitih primjera koji se razlikuju i koji bi trebali potvrditi ispravnost rada metode aukcije. Na kraju je zaključeno da metoda, uz sve svoje prednosti i mane, pruža optimalan raspored zadataka.

Ključne riječi: aukcija, roboti, sustavi, više-robotski sustavi, organizacija, podjela poslova

Task allocation in multi-robot systems based on auction method

Abstract

Large plants such as factories, hospitals, warehouses etc. often use devices or robots that automatically do the works. Such plants often run into problems with organization and task allocation when using multi-robot systems. This bachelor thesis researches one of the methods that provides optimal task scheduling - the auction method. The main issue is addressed first, then theoretical settings of algorithms and program implementation. The results of the developed method are presented on 5 different examples that differ and which should confirm the correctness of the auction method. In the end, it was concluded that the method, with all its advantages and disadvantages, provides an optimal schedule of tasks.

Keywords: auction, robots, systems, multi-robot systems, organization, task allocation