University of Zagreb

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Barbara Arbanas Pascoal Ferreira

DECENTRALIZED MISSION PLANNING FOR HETEROGENEOUS ROBOTIC TEAMS BASED ON HIERARCHICAL TASK REPRESENTATION

DOCTORAL THESIS

Zagreb, 2022

University of Zagreb

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Barbara Arbanas Pascoal Ferreira

DECENTRALIZED MISSION PLANNING FOR HETEROGENEOUS ROBOTIC TEAMS BASED ON HIERARCHICAL TASK REPRESENTATION

DOCTORAL THESIS

Supervisors:
Professor Stjepan Bogdan, PhD
Professor José Ramiro Martínez de Dios, PhD

Zagreb, 2022

University of Zagreb

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Barbara Arbanas Pascoal Ferreira

DECENTRALIZIRANO PLANIRANJE MISIJA ZA HETEROGENE ROBOTSKE TIMOVE TEMELJENO NA HIJERARHIJSKOM PRIKAZU ZADATAKA

DOKTORSKI RAD

Mentori:

Prof. dr. sc. Stjepan Bogdan

Prof. dr. sc. José Ramiro Martínez de Dios

Zagreb, 2022.

DOCTORAL THESIS is written at the University of Zagreb, Faculty of Electrical Engineering and Computing, Department of Control and Computer Engineering.

Supervisors: Professor Stjepan Bogdan, PhD and Professor José Ramiro Martínez de Dios, PhD

Doctoral thesis has 139 pages

Dissertation No.:

About Supervisors

Stjepan Bogdan, PhDEE is Full Professor at the Laboratory for Robotics and Intelligent Control Systems (LARICS - `https://larics.fer.hr/larics`), Department on Control and Computer Engineering, Faculty of Electrical Engineering and Computing, University of Zagreb, where he teaches several courses in robotics and automation. His research interests include autonomous systems, aerial robotics, multi-agent systems, intelligent control systems, bio-inspired systems, and discrete event systems. He is a co-author of 4 books and has published more than 200 scientific papers. He was the Principal Investigator and a researcher on 30 national and international scientific projects. Among others, he was the Principal Investigator of 2 projects financed by the Croatian Science Foundation (HRZZ), 2 bilateral Chinese-Croatian projects, 1 project funded by Air Force Office of Scientific Research, and 1 NATO SpS project. Currently, he is the coordinator of H2020 project AeroTwin. As the researcher, Prof. Bogdan has participated in 4 EU FP7 projects (ACROSS, EC-SAFEMOBIL, EuRoC, and ASSISI_bf), 5 H2020 projects (CROBOHUB, ACROSS, RoboCom++, subCULTron, and ENDORSE), and 1 HRZZ project (ADORE). Currently, he is involved in 4 H2020 projects (Watchplant, ENCORE, AeroWind, AerialCore), 4 EU Structural Funds projects (HEKTOR, ASAP, EKOKOMVOZ and VirtualUAV), and 1 HRZZ project (Specularia). The list with detailed descriptions of the past and the current projects is available at `https://larics.fer.hr/larics/scientific_projects`.

He served as an Associate Editor of IEEE Transactions on Automation Science and Engineering. Currently, he is an Associate Editor of several scientific journals. He served as a PC member and AE of major control and robotics conferences, and was the Program Chair of IEEE ISIC2011, and General Chair of IEEE MSC2012, 2018 MED Conference, IEEE 2021 AIRPHARO Workshop, and Honorary Chair of IEEE ICUAS2021. He is IEEE senior member, and an appointed member of IEEE Tech. Comm. on Intelligent Control and IEEE Tech. Comm. on Aerial Robotics. He was representative of Croatia at European Union Control Association (EUCA) and in H2020 Program Committee on Space. He is Program Committee member of Horizon Europe Cluster 4 – Digital, Industry and Space.

He spent 1996/1997 as a Fulbright scholar at Automation and Robotics Research Institute, The University of Texas at Arlington, under the supervision of Prof. Frank Lewis. He is a recipient of Bronze Plaque "Josip Lončar" (1990); Silver Plaque "Josip Lončar" (1993); The Croatian Society of University Teachers' "Best Young Scientist" award (2000); "Science Award" by the FER Faculty Council (2013); "Fran Bošnjaković" award by the University of Zagreb (2015).

Prof. José Ramiro Martínez de Dios finished his PhD in Telecommunication Engineering in 2001 and is a Full Professor at the University of Seville since 2017. He is a member of the GRVC Robotics Lab Sevilla (`https://grvc.us.es/`) since 1995. His R&D activities are focused on aerial robot perception, multi-robot cooperation, robot localization and mapping, sensor fusion, robot-sensor network cooperation, and Internet of Things.

In these topics, he has coordinated 7 R&D projects (5 EU-funded projects and 1 RETOS Spanish National R&D project) and has participated in other 56 R&D projects, including 18 projects funded by the European Commission in FPIV, FPV, FPVI, FP7, and H2020 Programmes and ERC Advanced Grant, and 8 Spanish National R&D Plan projects. Details of his projects are available at `https://investigacion.us.es/sisius/sis_showpub.php?idpers=1030`.

He has authored or co-authored >140 publications on these topics including 41 papers in JCR-index journals (18 Q1), 12 papers in IEEE ICRA/IROS, 4 full books (3 of them in Springer), and 14 book chapters (not originated by publications in conferences). His publications have received a total of >3760 cites (Scholar) and has H=29 (Scholar). Details of his publications are available at `https://scholar.google.com/citations?user=U3Ofd-wAAAAJ&hl=es`.

He has supervised 3 PhD Theses and 5 under supervision. He is a member of the Editorial Board of >7 JCR-indexed journals, Associated Editor of IEEE RA-L, and has been a TPC member in >60 conferences and workshops. He has organized >12 conferences and workshops. He has been a recipient or co-recipient of 5 international awards, including the Second EUROP/EURON Technology Transfer Award (2010) and the "Best Drone-based Solution" Award in the 1st EU Drone Awards in 2017. He has been a member of the Director Board of the Robotics Group from CEA (Spanish Section of IFAC) between 2004 and 2012 and Responsible for the White Book on Robotics between 2004 and 2008.

J. Ramiro Martínez-de Dios led 7 technology transfer contacts and participated in other 25 technology transfer contacts to 13 companies such as AIRBUS, ITURRI, BOEING RESEARCH AND TECHNOLOGY EUROPE, NAVANTIA, or IBERDROLA, among others, generating industrial products used such as: application of aerial robots to inspection (IBERDROLA, INGEMONT, DIAGNOSTIQA), aircraft networking (Boeing, TTI Norte), industrial IoT systems (AIRBUS, CLEVER), tracking using visual and infrared cameras, early autonomous detection of forest fires (Navantia), among others.

O mentorima

Dr. Stjepan Bogdan redoviti je profesor u Laboratoriju za robotiku i inteligentne sustave upravljanja (LARICS - https://larics.fer.hr/larics), Zavod za automatiku i računalno inženjerstvo, Fakultet elektrotehnike i računarstva Sveučilišta u Zagrebu, gdje predaje nekoliko kolegija iz robotike i automatizacije. Njegovi istraživački interesi uključuju autonomne sustave, zračnu robotiku, višeagentske sustave, inteligentne sustave upravljanja, biološki inspirirane sustave i sustave s diskretnim događajima. Koautor je 4 knjige i objavio je više od 200 znanstvenih radova. Bio je glavni istraživač i istraživač na 30 domaćih i međunarodnih znanstvenih projekata. Između ostalih, bio je glavni istraživač na 2 projekta financirana od strane Hrvatske zaklade za znanost (HRZZ), 2 bilateralna kinesko-hrvatska projekta, jednom projektu financiranom od Ureda za znanstvena istraživanja Zračnih snaga i jednom NATO SpS projektu. Trenutno je koordinator H2020 projekta AeroTwin. Kao istraživač, prof. Bogdan sudjelovao je u 4 EU FP7 projekta (ACROSS, EC-SAFEMOBIL, EuRoC i ASSISI_bf), 5 projekta H2020 (CROBOHUB, ACROSS, RoboCom ++, subCULTron i ENDORSE) i 1 HRZZ projektu (ADORE). Trenutno je uključen u 4 projekta H2020 (Watchplant, ENCORE, AeroWind, AerialCore), 4 projekta strukturnih fondova EU (HEKTOR, ASAP, EKOKOMVOZ i VirtualUAV) i 1 projekt HRZZ (Specularia). Popis s detaljnim opisima prošlih i aktualnih projekata dostupan je na `https://larics.fer.hr/larics/sciScientific_projects`.

Bio je pomoćni urednik časopisa IEEE Transactions on Automation Science and Engineering. Trenutno je suradnik urednik nekoliko znanstvenih časopisa. Bio je član programskog odbora i pomoćni urednik velikih konferencija s područja robotike i kontrole te je bio programski predsjedatelj IEEE ISIC2011 i generalni predsjedatelj IEEE MSC2012, 2018. MED konferencije, IEEE 2021 AIRPHARO radionice i počasni predsjedatelj IEEE ICUAS2021. On je stariji član IEEE-a i imenovan je članom IEEE tehničkog odbora o inteligentnoj kontroli i IEEE tehničkog odbora o zračnoj robotici. Bio je predstavnik Hrvatske u Udruzi za kontrolu Europske unije (EUCA) i u Programskom odboru H2020 za svemir. Član je Programskog odbora Horizon Europe Cluster 4 - Digital, Industry and Space.

Školsku godinu 1996./1997. proveo je kao Fulbrightov stipendist na Institutu za automatizaciju i robotiku Sveučilišta Texas u Arlingtonu, pod nadzorom prof. Franka Lewisa. Dobitnik je brončane plakete "Josip Lončar" (1990); srebrne plakete "Josip Lončar" (1993); nagrade "Najbolji mladi znanstvenik" Hrvatskog društva sveučilišnih nastavnika (2000); nagrade "Nagrada za znanost" Fakultetskog vijeća FER -a (2013); nagrade "Fran Bošnjaković" Sveučilišta u Zagrebu (2015).

Prof. José Ramiro Martínez de Dios doktorirao je telekomunikacijsko inženjerstvo 2001. godine, a redovni je profesor na Sveučilištu u Sevilli od 2017. Član je GRVC Robotics Lab Sevilla (`https://grvc.us.es/`) od 1995. Njegove istraživačko-razvojne aktivnosti usmjerene su na percepciju zračnih robota, suradnju višerobotskih sustava, lokalizaciju i mapiranje, fuziju senzora, suradnju robotskih i senzorskih mreža te Internet stvari.

U tim je temama koordinirao 7 projekata istraživanja i razvoja (5 projekata financiranih od strane EU-a i 1 španjolski nacionalni projekt istraživanja i razvoja RETOS) te je sudjelovao u drugih 56 projekata istraživanja i razvoja, uključujući 18 projekata financiranih od Europske komisije kroz FPIV, FPV, FPVI, FP7 i H2020 programe te ERC Advanced Grant te 8 projekata španjolskog Nacionalnog plana istraživanja i razvoja. Detalji o njegovim projektima dostupni su na `https://investigacion.us.es/sisius/sis_showpub.php?idpers=1030`.

Napisao je ili koautor >140 publikacija na ove teme, uključujući 41 rad u časopisima JCR indeksa (18 Q1), 12 radova u IEEE ICRA/IROS, 4 cjelovite knjige (od toga 3 u Springeru) i 14 poglavlja u knjigama (nisu nastale objavljivanjem na konferencijama). Njegove su publikacije primile ukupno >3760 citata i ima H=29 na Google scholar repozitoriju. Pojedinosti o njegovim publikacijama dostupne su na `https://scholar.google.com/citations?user=U3Ofd-wAAAAJ&hl=es`.

Vodio je 3 doktorske disertacije i trenutno vodi 5. Član je uredničkog odbora >7 časopisa indeksiranih prema JCR-u, pridruženi je urednik IEEE RA-L, a bio je i član tehničkog programskog odbora na >60 konferencija i radionica. Organizirao je >12 konferencija i radionica. Dobitnik je ili suprimitelj 5 međunarodnih nagrada, uključujući Drugu nagradu EUROP/EURON za transfer tehnologije (2010.) i nagradu "Najbolje rješenje temeljeno na bespilotnim letjelicama" na Prvim nagradama EU za bespilotne letelice 2017. Bio je član upravnog odbora za robotiku iz CEA-e (španjolski odjel IFAC-a) između 2004. i 2012. i odgovoran je za “Bijelu knjigu” o robotici između 2004. i 2008. godine.

J. Ramiro Martínez-de Dios vodio je 7 kontakata za prijenos tehnologije i sudjelovao u drugih 25 kontakata za prijenos tehnologije u 13 tvrtki kao što su AIRBUS, ITURRI, BOEING RESEARCH AND TECHNOLOGY EUROPE, NAVANTIA ili IBERDROLA, među ostalim, proizvodeći industrijske proizvode koji se koriste: primjena zračnih robota za inspekciju (IBERDROLA, INGEMONT, DIAGNOSTIQA), umrežavanje zrakoplova (Boeing, TTI Norte), industrijske IoT sustave (AIRBUS, CLEVER), praćenje vizualnim i infracrvenim kamerama, rano autonomno otkrivanje šumskih požara (Navantia), među ostalima.

To my family.

For their continuous support.

Acknowledgements

If we knew what it is we were doing,
it would not be called research.
Would it?
—Albert Einstein

The work presented in this dissertation is the result of years of research and would not be the same without the help of many of my colleagues and friends. Research is often a collaborative effort involving many people who can share ideas and come to new and interesting conclusions through trial and error.

First, I would like to thank my thesis advisor, Prof. Stjepan Bogdan, for supporting me over many years and giving me the opportunity to freely express and explore my ideas and follow my interests. He and Asst. Prof. Tamara Petrović have given me invaluable advice and perspective on my work and guided the progress of my research. Many thanks to my second supervisor, Prof. José R. Martínez de Dios, for his careful supervision and involvement in formulating some of the most important contributions to this work.

Next, I would like to thank my colleagues in the LARICS laboratory for creating a pleasant and productive work environment. We have been through many trials together and have always managed to motivate each other and move forward in our respective research. Many of the results presented in this thesis are the result of joint efforts in many research projects and would not have been possible without my colleagues. They have not only been great colleagues, but some of them have become very good friends for life.

However, all of this would not have been possible without the love and support of my family. I am especially grateful to my parents for allowing me to choose my own path and follow my passions. They gave me a secure base on which to develop according to my desires, and for that I am forever grateful.

Finally, I would like to thank my husband and my best friend for all the support and understanding he has given me over the years. He has been my greatest supporter and advisor, both personally and professionally, and his unconditional love has enabled me to build my confidence and reach my full potential.

In Zagreb, February 11, 2022

Abstract

This thesis focuses on the planning and coordination of cooperative missions for heterogeneous MRS. This complex problem consists of mission decomposition selection (the question *what do we do?*), task allocation (the question *who does what?*), and task scheduling (the question *how do we arrange tasks in time?*), which are often summarized under the common term mission (task) planning. Mission planning can be viewed as an optimization problem that attempts to find the most appropriate way to execute a mission, given certain criteria. Overlaid on this process is a set of coordination mechanisms that ensure timely and coordinated planning and execution of tasks between multiple individual robots.

This thesis proposes a distributed, multi-stage optimization method for planning complex missions for heterogeneous multi-robot teams. This class of problems includes tasks that can be executed in different ways and are associated with cross-schedule dependencies that constrain the schedules of the different robots in the system. In nature, these are NP-hard problems, and obtaining exact solutions is computationally intractable. The approach proposed in this thesis involves a multi-objective heuristic search of the mission model, represented as a hierarchical tree that defines the mission goal. This process identifies multiple favorable ways to accomplish the mission (task decomposition selection), which feed directly into the next phase of the method – task allocation and scheduling.

For task allocation and scheduling, a distributed algorithm is proposed for missions where the tasks of different robots are tightly coupled with temporal and precedence constraints. The approach is based on representing the problem as a variant of the Vehicle Routing Problem (VRP). The solution is found using a distributed metaheuristic algorithm based on evolutionary computation (CBM-pop). Such an approach allows for fast and near-optimal allocation and can therefore be used for online rescheduling in case of task changes. Simulation results show that the approach has better computational speed and scalability without loss of optimality compared to state-of-the-art distributed methods.

Finally, this work defines a decentralized coordination framework for heterogeneous multi-robot systems (GEM). The framework includes coordination mechanisms that ensure coordinated mission planning and execution. It also implements a complete software infrastructure that interacts with the specified hierarchical task model. It is domain-independent and supports different missions for heterogeneous multi-robot systems, as long as they conform to the mission specification model.

The proposed solutions are thoroughly tested in a realistic simulation environment as well as in real experiments using heterogeneous multi-robot systems. The proposed algorithms are directly compared with similar methods in the literature and the results show that our method can handle more complex missions with good scaling properties.

We provide suboptimal solutions in fast computation times, which makes the proposed solutions particularly suitable for real-world robotics applications.

The following original scientific contributions are achieved in this thesis:

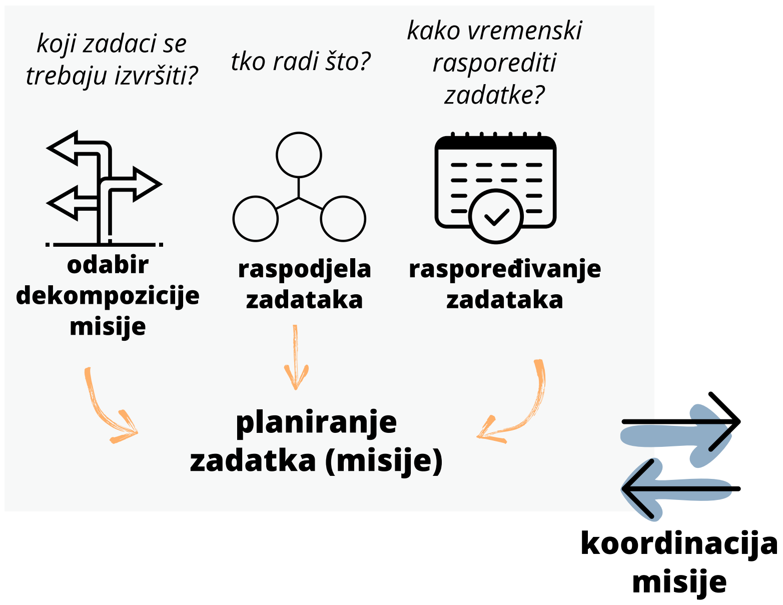
- A framework for decentralized task allocation, scheduling, and coordination of heterogeneous robotic teams based on hierarchical task representation.
- A method for distributed task allocation and scheduling for heterogeneous robotic team missions with cross-schedule task dependencies.
- A method for distributed mission decomposition selection for heterogeneous robotic team missions with complex task dependencies.

Keywords: Multi-Robot Planning, Multi-Robot Coordination, Task Allocation, Task Scheduling, Distributed Optimization, Multi-Robot Systems

Sažetak

Decentralizirano planiranje misija za heterogene robotske timove temeljeno na hijerarhijskom prikazu zadataka.

Znanstveno područje kooperativnih višerobotskih sustava (engl. *multi-robot system*, MRS) posljednjih godina uvelike zaokuplja pozornost istraživača. Razlog tome je što višerobotski sustavi mogu obavljati određene poslove učinkovitije od jednog robota, a neke zadatke nije ni moguće ostvariti korištenjem samo jednog robota. Nadalje, višerobotski sustavi imaju prednosti kao što su povećanje tolerancije na moguće greške robota, mogućnost fleksibilnijeg izvođenja zadataka te raspodijeljeno istraživanje i djelovanje. Dodatno, uvođenje heterogenosti, gdje svaki robot ima različite sposobnosti, dovodi do drugih zanimljivih implikacija na sustav upravljanja i omogućuje zanimljivo zajedničko ponašanje timova robota. Ipak, glavni je izazov osigurati robustan i inteligentan sustav upravljanja kako bi agenti mogli međusobno komunicirati i koordinirati se u izvršavanju zajedničkih misija. Stoga su razvoj sposobne upravljačke arhitekture, komunikacijske infrastrukture i sustava planiranja današnji glavni problemi istraživanja ovog područja.


koji zadaci se trebaju izvršiti?
tko radi što?
kako vremenski rasporediti zadatke?
odabir dekompozicije misije
raspodjela zadataka
raspoređivanje zadataka
planiranje zadatka (misije)
koordinacija misije

Ilustracija problema planiranja i koordinacije misije višerobotskih sustava.

Fokus ovog rada je **planiranje i koordinacija kooperativnih misija** za heterogene višerobotske sustave. Ovaj složeni problem sastoji se od odabira dekompozicije misije (engl. *task decomposition selection*; odgovara na pitanje *koji zadaci se trebaju obaviti?*), dodjele zadataka (engl. *task allocation*; odgovara na pitanje *tko radi što?*) te vremenskog raspoređivanja zadataka (engl. *task scheduling*; odgovara na pitanje *kako vremenski*

rasporediti zadatke?), koji se često sažimaju pod zajedničkim pojmom *planiranje misije (zadatka)* (engl. *mission (task) planning*). Planiranje misije može se promatrati kao problem optimizacije s ciljem pronalaska najprikladnijeg načina izvršenja misije prema zadanim kriterijima. Skup koordinacijskih mehanizama nadgleda ovaj proces te osigurava pravovremeno i koordinirano planiranje i izvršavanje zadataka više pojedinačnih robota. Ilustracija zadanog problema prikazana je na slici iznad.

U ovom radu misije predstavljene kao velike hijerarhije zadataka podvrgnute su optimizacijskom postupku u dva koraka. U prvom koraku provodi se brzo i učinkovito heurističko pretraživanje grafa misije koje pronalazi nekoliko obećavajućih alternativnih načina za izvršenje misije (procedura odabira dekompozicije zadataka). Zatim se postupak raspodjele i raspoređivanja zadataka primjenjuje na nekoliko najbolje rangiranih alternativa. Na temelju zadanih kriterija, najbolje cjelokupno rješenje je izlaz kao konačni raspored koji najbolje zadovoljava cilj misije. Tijekom raspodjele i raspoređivanja zadataka, problem se prikazuje kao varijanta problema raspoređivanja vozila (engl. *Vehicle Routing Problem, VRP*). Pritom je definiran *generički model problema planiranja zadataka* koji se može primijeniti na probleme iz različitih domena sustava s više robota.

Opisani postupak optimizacije ugrađen je u koordinacijski okvir za planiranje misije s više robota. Okvir uključuje mehanizme koordinacije koji osiguravaju koordinirano planiranje i izvršavanje misija. Također implementira kompletnu softversku infrastrukturu koja se povezuje s hijerarhijskim modelom zadataka. Ovaj pristup pojednostavljuje primjenu planera na različite domene, budući da se unutar ove dobro provjerene infrastrukture za koordinaciju i planiranje trebaju implementirati samo specifične misije.

Disertacija je podijeljena u sljedeća poglavlja:

- Introduction;
- Background and Related Work;
- Decentralized Coordination of Heterogeneous Robotic Teams;
- Definition and Modeling of Task Planning Problems;
- Mission Planning Solution Approach;
- Results and Discussion;
- Conclusion.

U uvodu je dan pregled područja planiranja misija u heterogenim robotskim sustavima i opis problema koji će se razmatrati u doktorskom radu. U glavnim crtama opisana je struktura rada te je dan sažetak znanstvenih doprinosa.

U drugom poglavlju dana je temeljita analiza postojećih rješenja za planiranje misija i koordinaciju u heterogenim višerobotskim sustavima. Također je definiran problem planiranja misija te njegova klasifikacija. U području planiranja misija postoji nekoliko klasa problema, ovisno o svojstvima robota i zadataka. Najčešća podjela definirana je s obzirom na četiri parametra. Prvo, definirani su sustavi s robotima sposobnima obavljati jedan zadatak u nekom trenutku (engl. *single-task robots*, ST) od onih koji mogu obavljati više zadataka istovremeno (engl. *multi-task robots*, MT). Zadaci se na sličan način dijele na one za čije je izvođenje potreban jedan robot (engl. *single-robot tasks*, SR), te na zadatke u čijem izvršavanju sudjeluje više robota (engl. *multi-robot tasks*, MR). Treća podjela razmatra horizont planiranja i probleme dijeli na one u kojima se zadaci dodjeljuju u trenutku pristizanja u sustav (engl. *instantaneous assignment*, IA) te na probleme u kojima se u planiranju misije razmatraju trenutno dostupni, ali i budući zadaci (engl. *time-extended assignment*, TA). Za probleme klase TA za svakog robota izrađuje se raspored izvršavanja zadataka. Naposljetku, posljednji kriterij tiče se složenosti zadataka te se problemi dijele na četiri kategorije: bez međuovisnosti (engl. *no dependencies*, ND), s međuovisnostima unutar rasporeda pojedinog robota (engl. *in-schedule dependencies*, ID), s međuovisnostima između rasporeda različitih robota (engl. *cross-schedule dependencies*, XD), te na zadatke s kompleksnim međuovisnostima (engl. *complex dependencies*, CD). Kategorija CD uključuje zadatke s međuovisnostima između rasporeda različitih robota za zadatke s više mogućih načina izvođenja (s više različitih dekompozicija). U ovoj kategoriji problema kvaliteta kojom neki agent može izvesti zadatak ovisi o rasporedima drugih agenata u sustavu, i to je određeno odabranim načinom izvođenja pojedinih zadataka.

Od postojećih rješenja problema planiranja misija, ovo poglavlje analizira sustave temeljene na modelima, gdje su sustavi kao cjelina ili pojedini roboti opisani modelom koji upravlja njihovim ponašanjem. Ovi se modeli obično kreiraju prije izvođenja misije (engl. *offline*) i usmjeravaju radnje robota na temelju informacija koje primaju tijekom izvođenja misije kako bi se optimizirala ukupna korisnost misije. Drugo, promatra se eksplicitniji tip upravljanja višerobotskih sustava koji je temeljen na planiranju misije. Ovdje su eksplicitno modelirane misije podvrgnute proceduri optimizacije koja odlučuje o najboljem tijeku akcije za cijeli sustav ili određene robote. Planiranje se može izvršavati prije ili tijekom izvršavanja misije (*offline* ili *online*) i, ovisno o specifičnom pristupu, može biti manje ili više reaktivno na različite poremećaje ili neizvjesnosti u izvršavanju misije.

U trećem poglavlju opisan je model za hijerarhijsko predstavljanje zadataka, s detaljnim opisom elemenata modela i prikazom dekompozicije, odnosa i evaluacije zadataka. Glavna premisa hijerarhijskog predstavljanja zadataka je *dekompozicija zadataka*, gdje se veliki i potencijalno složeni zadaci postupno razlažu na jednostavnije, sve do razine elementarnih, izvedivih zadataka (akcija). Ovaj prikaz pruža bolji pregled misije i odnosa između zadataka te uvelike pojednostavljuje specifikaciju misije. Štoviše, bogata ekspresivnost formulacije misije omogućuje definiranje složenih odnosa zadataka i dekompozicija, a time i primjenjivost u različitim područjima. Ovaj rad vodi se općim značajkama TÆMS (engl. *Task Analysis, Environment Modeling and Simulation*) modela prikaza zadataka, s prilagodbama specifičnima koordinacijskom okviru razvijenom u ovoj disertaciji.

Također je opisan razvijeni okvir za decentralizirano planiranje zadataka, raspoređivanje zadataka i koordinaciju heterogenih robotičkih timova. Razvoj okvira inspiriran je GPGP (engl. *Generalized Partial Global Planning*) okvirom, koje se često koristi u sinergiji sa strukturom zadatka TÆMS. GPGP služi kao putokaz za distribuiranu koordiniranu kontrolu sustava s više agenata dajući opće smjernice za formulaciju koordinacijskih mehanizama i strukture modula. Na temelju identificiranih glavnih slabosti GPGP pristupa, u ovom radu je predloženo rješenje koje bolje odgovara potrebama i mogućnostima modernih robotskih sustava. Iako teoretski vrlo zanimljivi, neki od mehanizama GPGP-a uvode nepotrebna odstupanja od optimalnog rješenja rasporeda zadataka, uz pretpostavku smanjenja dostupnosti informacija svim robotima u sustavu. Ideja zadržavanja samo lokalnih pogleda na misije robota, a zatim dijeljenja potrebnih informacija može se lako ublažiti dopuštanjem svim robotima da zadrže globalnu strukturu misije. Mogućnosti pohrane podataka i komunikacije eksponencijalno su se razvile u modernim računalnim sustavima otkako je stvoren GPGP, tako da je drugačiji pristup lako moguć. Povrh toga, planer definiran u sklopu GPGP okvira, koji planira rasporede lokalno za pojedine robote te ih potom usklađuje s drugim robotima u sustavu, može se zamijeniti modernim distribuiranim rješenjima za raspoređivanje zadataka koja istovremeno rješavaju dodjelu i raspoređivanje zadataka.

Stoga je razvijen okvir GEM (engl. *GEneric Multirobot mission coordination and planning based on hierarchical task representation*). Okvir je implementiran u ROS-u (engl. *Robot Operating System*) i može se lako primijeniti na različite višerobotske timove. Osnovna ideja odvajanja domenski specifičnih i generičkih modula održava se u GEM arhitekturi. Za specifične aplikacije potrebno je implementirati module *Task Assessor* (procjenitelj zadataka) i *Task Executor* (izvršitelj zadataka). Okvir pruža upute i osnove za specifične implementacije u klasama predlošcima (engl. *template class*) koje osiguravaju besprijekornu integraciju s ostatkom okvira. Korisnici trebaju pružiti svoje specifične implementacije funkcija kako bi podržali željenu novu funkcionalnost za različite misije. Moduli *Mission Planner* (planer misije) i *Mission Coordinator* (koordinator misije) implementirani su i zajednički za različite primjene. Korisnik može zamijeniti zadani planer prilagođenom implementacijom ukoliko želi uvesti nove načine planiranja. Oba modula su dizajnirana s prilično jednostavnim sučeljem za ostatak sustava. Koordinator misije komunicira s drugim robotima u sustavu i koordinira cijeli proces planiranja i izvršenja. Ovaj modul je jednostavniji od GPGP koordinatora jer GEM rješenje ne definira toliko složene koordinacijske mehanizme.

Četvrto poglavlje sadrži objašnjenje načina matematičkog modeliranja problema planiranja zadataka za XD klasu problema, koji zahvaljujući odabranom pristupu ima svojstva generičkog modela zasnovanog na VRP (engl. *Vehicle Routing Problem*) paradigmi. U ovom poglavlju prikazuje se osnovni problem usmjeravanja vozila i daje njegov odnos prema problemu planiranja zadatka. Poveznica između dva modela (planiranje misija i problem usmjeravanja vozila) razmatrana je na visokoj razini apstrakcije, iako u nekim primjenama koje uključuju scenarije prijevoza odnos može biti očitiji. U modeliranju planiranja zadataka kao varijante VRP-a koristi se pojam kupca (engl. *customer*) za jednostavne zadatke (akcije). Ova definicija uključuje samo izvedive elemente iz strukture zadatka robota, a svi se ostali zadaci moraju razložiti do razine akcije na kojoj je optimiziran problem usmjeravanja vozila. U paradigmi planiranja zadataka pojam vozila (engl. *vehicle*) izravno odgovara robotima. Uobičajeno za VRP paradigmu je da vozila krenu iz skladišta (engl. *depot*), poslužuju dodijeljene kupce duž rute i vrate se u skladište. U modelu planiranja misije kao VRP, pojam skladišta izjednačava se s početnim stanjem robota. Za razliku od vozila u tipičnom VRP problemu, kod planiranja misija robot se ne treba vratiti u inicijalno stanje nakon izvršenja misije. Ova varijanta VRP problema naziva se otvoreni VRP (engl. *open VRP*). U modelu su definirani različiti parametri koji utječu na planiranje misije. Za svaki zadatak u sustavu definirano je trajanje (engl. *duration*), energetski zahtjev (trošak, engl. *cost*) te kvaliteta, ovisno o robotu koji ga izvodi. Dodatno, definirane su tranzicijske relacije, koje određuju vrijeme i trošak prelaska s jednog na drugi zadatak. Ovaj model dodatno definira ograničenja na izvođenje zadataka: sinkronizacijsko ograničenje (engl. *synchronization constraint*), koje omogućava istovremeno izvođenje dvije akcije, te ograničenje prioriteta (engl. *precedence constraint*), koje osigurava da se pojedine akcije izvedu prije nekih drugih akcija. Rješavanjem ovako definirane misije dobiva se raspored (engl. *schedule*) koji određuje redoslijed izvođenja akcija za svakog robota te vremenske trenutke početka i kraja njihovog izvršavanja.

U petom poglavlju dana su rješenja za planiranje misija s međusobnim ovisnostima zadataka, kao i složenim ovisnostima zadataka. Problemi klase XD (s međuovisnostima između zadataka različitih robota) modelirani su kao VRP problem definiran u prethodnom poglavlju te podvrgnuti distribuiranoj metaheurističkoj optimizaciji za pronalazak suboptimalnih rasporeda. Rješenje je inspirirano metaheuristikom zasnovanom na koaliciji (engl. *Coalition-Based Metaheuristic*), CBM, gdje više agenata organiziranih u koaliciju istovremeno istražuju prostor rješenja, surađuju i samoprilagođavaju se kako bi zajednički riješili zadani problem. Novost uvedena u ovaj algoritam je korištenje osnovnih principa distribuirane umjetne inteligencije (engl.*Distributed Artificial Intelligence*, DAI), učenja potkrijepljenjem (engl. *reinforcement learning*) te mimetičkog učenja (engl. *mimetic learning; mimetism*), koji ne samo da omogućuju agentima da uče iz svojih iskustava i sukladno tome prilagode svoje buduće ponašanje, već i da dijele znanje s drugim agentima u koaliciji. Osim naučenih ponašanja, agenti dijele i najbolja pronađena rješenja, tako da se na kraju svake iteracije algoritma dobiva najbolje globalno rješenje problema.

U potrazi za optimalnim rješenjem, svaki agent koristi nekoliko operatora koji se primjenjuju na trenutno rješenje. Operatori mogu biti intenzifikatori (engl. *intensificator*) ili diverzifikatori (engl. *diversificator*). Operatori intenzifikacije odnose se na procese poboljšanja kao što je lokalno pretraživanje, a operatori diverzifikatora odgovaraju postupcima generiranja, mutacije ili križanja, slično kao u genetskim algoritmima (engl. *generation, mutation, crossover*). Izbor operatora za primjenu nije potpuno stohastičan kao u genetskim algoritmima. Umjesto toga, određen je procesom odlučivanja koji koristi percipirano stanje i prošlo iskustvo za odabir najprikladnijeg operatora te koordinira postupke intenzifikacije i diverzifikacije. Odabir operatora temelji se na heurističkim pravilima. Ponašanje agenta u pretraživanju prilagođava se tijekom procesa optimizacije kroz individualno učenje potkrijepljenjem te mimetičko učenje. Ovi mehanizmi modificiraju pravila procesa odlučivanja na temelju rezultata iskustva prethodnih istraživanja. Iako svi agenti u koaliciji koriste isti skup operatera, mehanizmi učenja mogu u konačnici ispoljiti različite strategije.

Problem odabira dekompozicije zadatka klase CD (engl. *task decomposition selection*) uključuje pronalaženje podskupa akcija i zadataka koje treba izvesti, s najvećim izgledima za ostvarenje rasporeda blizu optimalnih. U ovom koraku postupka, heuristički algoritam pretraživanja stabla koristi se za brzo generiranje alternativnih podskupova zadataka koji zadovoljavaju cilj misije. Složenost postupka pretraživanja grafa ovisi o strukturi stabla misije te odnosima između čvorova. Dodavanje više ograničenja pojednostavljuje proceduru jer uklanja neke od opcija za izvršavanje zadataka. Međutim, kombinatorna eksplozija može dovesti do faktorijelne složenosti postupka generiranja alternativnih načina izvođenja zadatka za misije bez ikakvih odnosa čvorova.

Proces generiranja alternativa zadatka počinje u akcijskim čvorovima stabla misije i rekurzivno se izgrađuje, na kraju završavajući u korijenu stabla. Kako bi ukrotio potencijalnu kombinatornu eksploziju, postupak koristi metodu fokusiranja traženja rješenja uklanjanjem najgorih djelomičnih rezultata u svakom koraku procesa kako bi se problem učinio rješivim. Tijekom ovog postupka roboti koriste procijenjene vrijednosti za kvalitetu, trajanje i cijenu radnji, koje se određuju kao prosjek tih vrijednosti za sve robote koji mogu izvršiti radnju.

U šestom poglavlju prikazano je nekoliko studija s implementacijom predloženog rješenja na različite heterogene višerobotske sustave, uključujući ispitivanja robotskog sustava letjelica i autonomnih vozila u zadatku autonomne dostave paketa, decentraliziranu kontrolu simbiotskog tima letjelice i mobilnih vozila te koordinaciju autonomnog zračno-zemaljskog tima za zadatke automatizirane gradnje.

Zaključak provedenog istraživanja dan je u posljednjem sedmom poglavlju. Pokazano

je da oba rješenja (za XD i CD klasu problema) uključuju korištenje heuristika i aproksimacija za brzo dobivanje suboptimalnih rješenja za ove vrlo teške kombinatorne probleme. Oba ponuđena pristupa pronalaze rješenja jednako dobro ili bolje od najmodernijih pristupa iz literature, uz značajno ubrzanje optimizacijskog postupka.

Disertacijom je ostvaren sljedeći izvorni znanstveni doprinos:

1. Radni okvir za decentraliziranu dodjelu i vremensko raspoređivanje zadataka, te koordinaciju heterogenih robotskih timova temeljen na hijerarhijskom prikazu zadataka.
2. Metoda za distribuiranu dodjelu i vremensko raspoređivanje zadataka heterogenih robotskih timova za misije s međuovisnostima zadataka različitih rasporeda.
3. Metoda za distribuiran odabir dekompozicije misije heterogenih robotskih timova za misije sa složenim međuovisnostima zadatka.

Ključne riječi: planiranje misija višerobotskih sustava, koordinacija višerobotskih sustava, dodjela zadataka, raspoređivanje zadataka, distribuirana optimizacija, višerobotski sustavi

Contents

CHAPTER 1

Introduction

Cooperative *Multi-Robot Systems* (MRS) have received much attention in recent decades [1, 2]. The great interest in these systems stems from the considerable difficulty in establishing intelligent, coherent control of joint missions and the many advantages that they offer. Compared to a single robot, MRS can leverage the strengths of the participating robots to establish a more robust system that is more resilient to various issues, such as robot or sensor failures. Furthermore, the introduction of heterogeneity, where each robot has different capabilities, leads to other interesting implications for the control system and allows for intriguing collaborative behavior between robots [3]. The main challenge is to provide a robust and intelligent control system so that the agents can communicate and coordinate with each other to accomplish a mission given to them. Therefore, developing capable control architecture, communication, and planning system are the main problems discussed and solved among researchers [2].

This thesis focuses on the **planning and coordination of cooperative missions** for heterogeneous MRS. This complex problem consists of mission decomposition selection (the question *what do we do?*), task allocation (the question *who does what?*), and task scheduling (the question *how do we arrange the tasks in time?*), which are often summarized under the common term *mission (task) planning* [4]. Mission planning can be viewed as an optimization problem that attempts to find the most appropriate way to execute a mission according to given criteria. Overlooking this process is a set of coordination mechanisms that ensure timely and coordinated planning and execution of tasks between multiple individual robots. An illustration of the given problem is shown in Figure 1.1.

In this work, missions represented as large task hierarchies are subjected to a two-stage hierarchical optimization procedure. In the first step, we perform a fast and efficient heuristic search of the mission tree that finds several promising alternative ways to execute the mission (task decomposition selection procedure). Then, a task allocation and scheduling procedure is applied to several best-ranked alternatives to generate schedules

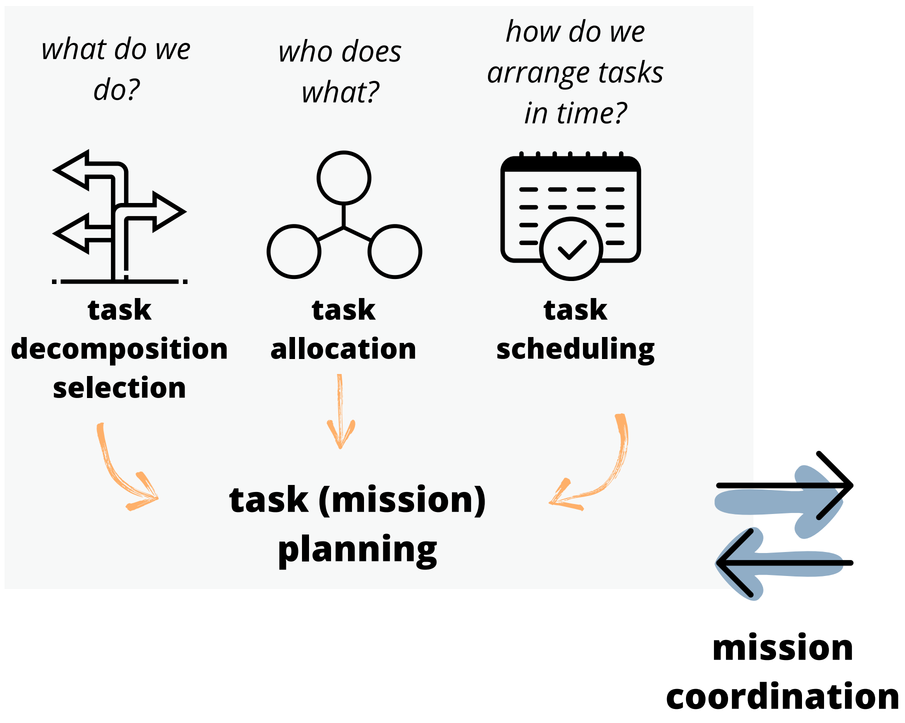


Figure 1.1: Illustration of mission planning and coordination problems of MRS.

for the given problem. Based on the given criteria, the best overall solution is output as the final schedule that best satisfies the mission objective.

During task allocation and scheduling, the problem is presented as a variant of *Vehicle Routing Problem* (VRP). In doing so, we define a *generic model of task planning problems* that can be applied to problems from different domains of multi-robot and multi-agent systems. The proposed solution acts as a domain-agnostic planner of problems that adhere to the specified model.

The described optimization procedure is embedded in a coordination framework for multi-robot mission planning. The framework includes coordination mechanisms that ensure coordinated planning and execution of missions. It also implements a complete software infrastructure that interfaces with the *Task Analysis, Environment Modeling and Simulation* (TÆMS) hierarchical task model. This approach simplifies the application of the planner to different domains, as only the specific missions need to be implemented within this well-tested coordination and planning infrastructure, as shown in the results section.

1.1 Problem Features

Some of the main features of mission planning and coordination problems considered in this thesis are given as follows:

- *Task decomposability.* Missions considered for multi-robot system planning consist of simple tasks (actions) that can be combined in various ways to form composite tasks. Therefore, each composite task can be decomposed to obtain a set of simple, executable tasks that a robot can perform.
- *Heterogeneity of robots and tasks.* We assume a multi-robot team in which each robot may have its own capabilities, which can differ from those of other robots in the system. Similarly, tasks may require different abilities in their execution, so that only a subset of robots can participate in them.
- *Inter-task ordering constraints.* A particular task may need to be performed before or at the same time as another task, resulting in precedence and synchronization constraints. These relationships can constrain the tasks of a single robot (intra-schedule) or multiple robots (inter-schedule or cross-schedule). Ordering constraints directly affect the schedules of specific robots and the allocation of tasks among robots.
- *Capacity constraints.* We assume that each task requires a certain amount of a resource to be executed, e.g., a battery, load capacity, or any other unit. Similarly, the robots have specified the maximum and minimum values for each resource at which the operations are still possible. If the resource is depleted (or above the allowed limits), tasks that require it cannot be executed.
- *Temporal constraints.* A task may have a time window within which it must be performed, and robots should adhere to it. This constraint can be hard or soft, depending on how strict it is. Soft constraints have a penalty for exceeding the time window, while hard constraints do not allow schedules with broken time window requirements.

1.2 Thesis Contributions and Overview

The objective of the proposed research is the synthesis of a complete planning system for distributed mission decomposition selection, task allocation, scheduling, and coordination of heterogeneous robotic teams based on hierarchical task representation for complex multi-robot missions.

Therefore, scientific contributions of this thesis are summarized as follows:

- **First.** A framework for decentralized task allocation, scheduling, and coordination of heterogeneous robotic teams based on hierarchical task representation.
- **Second.** A method for distributed task allocation and scheduling for heterogeneous robotic team missions with cross-schedule task dependencies.
- **Third.** A method for distributed mission decomposition selection for heterogeneous

robotic team missions with complex task dependencies.

The first contribution entails a comprehensive coordination framework for heterogeneous multi-robot teams. The mission-agnostic framework serves as a basis for mission planning systems that adhere to the given mission specification structure. An advantage of such a framework is that it enables faster and more reliable implementation by providing an architecturally stable foundation on which specific planners can build.

As a second contribution, we propose a novel model that unifies task planning models with a well-studied VRP. We define a mathematical model formulation for mission planning problems with cross-schedule dependencies. The cross-schedule dependencies we consider are precedence constraints and transitional dependencies (including the cost and time limitations of switching between any two actions). Besides generalizing the problem, another advantage of the proposed modeling is that it exposes task planning problems to a wide range of optimization techniques already available for VRP, thus advancing the state of the art in task planning. We propose a fast distributed metaheuristic approach to solve the described problem.

Finally, we address the problems with even higher complexity. By combining different methods and modeling defined in the previous contribution, we solve problems of task decomposition selection, task allocation, and scheduling. The current literature generally does not consider problems of such complexity. Our approach provides a fast suboptimal solution to this NP-hard [5] combinatorial problem and is suitable for real-world robotics applications.

Thesis Organization

The thesis is organized as follows. In the next chapter, we describe the problems of mission planning and coordination for MRS and review the current state of research. Next, in Chapter 3 we define the underlying hierarchical mission model and the development of the coordination framework for MRS (Contribution 1). Then, in Chapter 4, we present the mathematical modeling of the mission planning problems as a generic model based on the VRP paradigm. Next, Chapter 5 outlines the solution approach to the planning problem defined in the previous chapter (Contribution 2), and the solution to the CD class of problems (Contribution 3). In Chapter 6, we present the main results of the conducted research and discuss the findings. Finally, in Chapter 7, conclusions and insights are given for future work and further development of the proposed work.

CHAPTER 2

Background and Related Work

Research in cooperative MRS is of interest to various research groups, as it holds many interesting and useful potentialities for cooperative mission execution. A multi-robot system can be homogeneous or heterogeneous [2]. In *homogeneous* robot teams, the capabilities of the individual robots are identical (the physical structures do not have to be the same) [6, 7]. In *heterogeneous* robot teams, the capabilities of the individual robots are different, allowing them to specialize in certain tasks [8, 9].

In [10], the authors proposed a taxonomy for coordinated MRS as shown in Figure 2.1. At the top level, MRS can be classified into two groups – *cooperative* and *competitive*. A cooperative system consists of robots working together to accomplish a global task. Competition in MRS is a fairly unexplored area, as there are not many uses for a competitive multi-robot system. They are mostly studied from the perspective of competitive games, such as robot soccer.

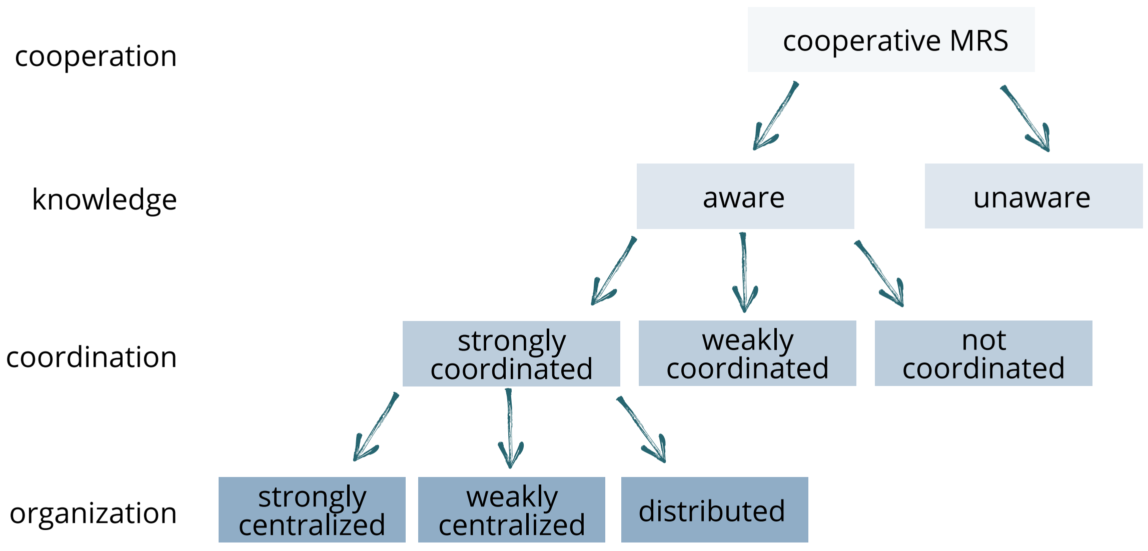


Figure 2.1: Classification of cooperative MRS focused on coordination.

The second category classifies systems based on the knowledge that each robot has about its peers. In this sense, robots can be either *aware* or *unaware*. The main advantage of unaware MRS is its simplicity, since no coordination and communication protocols need to be developed. However, this is also a disadvantage, as they are unaware of each other, they are unpredictable, and it is difficult to estimate the impact of individuals' actions on the overall utility of the system.

Next, aware MRS can be categorized according to the degree of coordination within the system. *Coordination* is defined as a cooperative act in which each robot takes into account the actions of the other robots in such a way that the whole becomes a coherent and high-performance team. How this property comes into play depends on the use of the coordination protocol, i.e., a set of rules that the robots must follow in their actions. In this sense, MRS can be divided into three main categories – *strongly coordinated*, *weakly coordinated* and *not coordinated*, depending on how much it relies on a coordination protocol. In this thesis we will focus exclusively on systems with some degree of coordination, since this is the area we are interested in. These systems can be described as *teams* that coordinate their actions to achieve a common goal.

The final refinement classifies strongly coordinated MRS based on the organization of the control architecture. Two main classes of protocols are distinguished: *centralized* and *distributed*. In the former, all control mechanisms are implemented on a central computing unit, and the robots simply follow the instructions given to them. Centralized MRS can be either strongly centralized, where the central unit is predefined and all computations are performed on it, or weakly centralized, where more than one robot can assume the role of the leader during the mission. In contrast, in distributed control systems, the processing is spread across multiple nodes. Various nodes can communicate and coordinate by passing messages. Within distributed architectures, there is a separate group – *decentralized* systems, where a decision is made across various nodes. Each node decides its behavior, and accumulated control outputs from all the nodes form the group behavior. Specific to decentralized systems is that no single node has complete system information.

Due to the global view of the mission, a centralized approach to multi-robot control can produce optimal or near-optimal plans [11, 12, 13]. Nevertheless, this architecture: 1) is typically used for a small number of robots and ineffective for large teams with more robots; 2) is not robust to dynamic environments or failures in communication and other uncertainties; 3) produces a highly vulnerable system in case of malfunction of the central control agent. In contrast, decentralized and distributed architectures usually exhibit higher reliability, flexibility, adaptability, and robustness. This makes them particularly suitable for field operations in dynamic environments where more robots are deployed with limited resources, even if the solutions they provide are often suboptimal. However, for agents to perform their actions in a distributed manner, appropriate coordination mechanisms must be developed.

Further, multi-robot coordination protocols can be *static* (also referred to as *offline*), where robots use predefined plans or rules for execution [14], or *dynamic* (also known as *online*), where plans are created during mission execution and respond to the state of the environment [15]. The static method can handle complex missions, but it cannot adapt to changes, so it is usually unsuitable for reactive real-time applications that require dynamic planning. The dynamic method, on the other hand, can perform in real-time very well, but has difficulties in handling more complex tasks.

2.1 Coordinated Control of Multi-Robot Systems

In the rest of the state of the art, we distinguish two types of coordinated control for MRS. First, in this section, we analyze *model-based systems*, where the systems as a whole or particular robots are described by a model that governs their behavior. These models are typically created offline and guide the robots' actions based on inputs they receive during the mission runtime in order to optimize the overall utility of the mission. Second, we observe in more detail a more explicit type of MRS control called *mission planning-based systems*. Here, the explicitly modeled missions are subjected to an optimization procedure that decides the best course of action for the entire system or particular robots. Planning can be done online or offline and, depending on the specific approach, can be more or less reactive to various disturbances or uncertainties in mission execution.

2.1.1 Model-based systems

In general, model-based control of MRS can be viewed as a *policy* that agents should implement to accomplish their assigned task once the complex tasks have been decomposed into subtasks [16]. Several frameworks have been proposed to model and solve decision-making problems, including logic controllers, game theory, and swarm intelligence. All of these approaches involve the *synthesis of controllers* that guide the coordinated behavior of a multi-robot team. The main goal is to model the behavior of robots for specific missions, with less explicit control. This in turn complicates the design of robot decision-making model, which increases computational complexity or, if simpler models are used, decreases the optimality of mission execution.

Logic-based controllers

Many of the current approaches rely on off-the-shelf automated reasoners based on, for example, Linear Temporal Logic (LTL) [17, 18, 19]. In [17], the authors propose a simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems. The main idea is to construct an offline team model used both for plan execution and for assigning specific subtasks to robots. This approach is particularly suitable for cases where task execution plans cannot be computed in advance, as for LTL-based planning of on-demand missions. The motivation for this approach comes from the use cases of service robots, e.g., in the office environment. Typical use cases are service tasks such as transporting documents or supplies, emptying waste bins, etc. The approach assumes a general model of the system containing a topological map of the environment and internal states of the robots.

A similar application can be seen in [20], where the authors task robots to provide delivery services in known locations, avoiding obstacles and dangerous regions. The proposed approach is the first reactive and abstraction-free LTL planning algorithm used for mission planning and coordination of multiple robots in unknown environments. The method adapts to the updated map of the environment as the robots explore new regions. It is abstraction-free, as it does not rely on creating abstractions of the robot dynamics. The algorithm is complete[*] under mild assumptions about the structure of the environment and sensor models.

Although these solutions represent a significant contribution to the theoretical synthesis of correct-by-design controllers, they often suffer from intense computational problems as well as the inability to quantify planner objectives and define complex task relationships. Recently, in [22], the authors proposed a new, highly scalable and asymptotically optimal algorithm for synthesising controllers from LTL specifications called STyLuS*. It is designed to solve complex temporal planning problems in large-scale multi-robot systems. Existing planning strategies with temporal logic specifications are based on graph search techniques applied to a product automaton constructed between robots. Here, the authors propose a more tractable sampling-based algorithm that incrementally builds trees that approximate the state space and transitions of the synchronous product automaton, avoiding the use of elaborate graph search techniques. However, the logic-based controllers are inherently limited by the expressivity of the underlying language and its ability to define complex missions.

Game-theoretic models

Game-theoretic models include partially observable stochastic games, which, because of partial observability of the world, are seen as sequential probabilistic games. A game unfolds over a finite or infinite sequence of stages. At each stage, all agents simultaneously

[*]If an algorithm is complete, it means that if at least one solution exists then the algorithm is guaranteed find a solution in a finite amount of time. [21]

select an action and receive a reward and observation. The objective, for each agent, is to maximize the expected sum of rewards it receives during the game. Whether agents compete or cooperate in seeking reward depends on their reward functions. Probabilistic games sub-classes include Partially Observable Markov Decision Processes (POMDPs) [23, 24]. The advantage of this approach is its inherent suitability for uncertain environments; however, the scalability problem makes them unsuitable for real-world applications with multiple robots and complex tasks.

Decentralized POMDPs (Dec-POMDPs) [25, 26, 27] are particularly widespread in MRS applications because their decentralized nature makes them particularly appropriate for multi-robot systems. Here, each agent chooses its action at each step based solely on locally observable information, resulting in each agent making an observation and the team receiving a joint reward. The joint reward function makes the problem cooperative, but the local views indicate decentralized execution.

The most common applications include formation keeping [26], environmental monitoring [28], and search and rescue [29]. These problems have relatively simple task domains and significant uncertainties in observing the state of the world and the internal conditions of the other robots. The problem of expanding the task domain (or the number of agents in the system) is prevalent in these models because the computational cost is too high to handle greater complexity. In addition, most methods require the model to be computed offline, making them incapable to find outcomes that were not considered in the original world and mission modeling.

Recently, some valuable online solutions have been reported [27]. The method, called DESPOT-α, outperforms all state-of-the-art online planning for POMDPs with large observation and state spaces. It uses the particle belief approximation and searches a determinized sparse belief tree. To cope with large observation spaces, partial policies are divided among many observations during online policy calculation. The method is further accelerated by CPU and GPU parallelization. The approach is evaluated on a complex simulation task where an autonomous vehicle drives between many pedestrians. However, the problems that this approach can address are still relatively simple from a high-level mission planning perspective.

Swarm intelligence-based systems

Inspired by social animals, swarm intelligence models many decentralized, cooperative, and autonomous agents [30]. Such systems are primarily characterized by self-organized and distributed behavior of locally interacting and locally aware agents. Given the number of units, swarm-based systems are robust, flexible, and fault-tolerant [31]. Their simple design makes them computationally inexpensive and particularly scalable. However, the robots in such systems tend to be homogeneous, which severely limits their applications [32]. Furthermore, due to the simplicity of the robot design, the tasks assigned to the robot swarms are generally of lower complexity.

Many optimization algorithms are inspired by swarm intelligence, where biological processes are modelled in search of optima. Particle Swarm Optimization (PSO) [33] is a stochastic optimization technique based on swarm behavior. The PSO algorithm simulates the social behavior of animals, including insects, flocks of birds, and schools of fish. These swarms search for food together, and each member of the group changes the search pattern according to its own and other members' learning experiences. Similarly, the pigeon-inspired optimization algorithm [34] relies on the magnetic field, the sun, and landmarks to achieve path planning. Bee colony optimization [35] is based on the behavior of bees and relies on direct communication between agents.

2.1.2 Mission planning

As mentioned earlier, in contrast to model-based solutions for multi-robot mission coordination, there is mission planning approach. In this approach, the missions of MRS are explicitly represented and subjected to various optimization procedures. As described earlier, planning consists of mission decomposition selection, task allocation, and scheduling.

There are many approaches to solving this complex problem and especially its subproblems. Solutions include various models of mission representation, search algorithms, and computing architectures. Since our approach concerns the mission planning problem, we devote more attention to it in the following sections. In Section 2.2, we outline the general taxonomy of mission planning problems and then present the state of the art.

2.2 Mission Planning Problems Taxonomy

Before describing the classification of mission planning problems, it is important to define the different types of tasks in terms of their complexity. Intuitively, there are simple, elemental tasks, i.e., actions that can be performed by a single robot, and that are the building blocks of any mission. Other tasks can be decomposed into multiple subtasks or actions. These are called compound tasks, provided there is a single fixed way to decompose the task into subtasks. Different parts of a compound task may be assigned to different agents. Finally, a complex task is one for which there are multiple ways to decompose it.

All three types are illustrated in Figure 2.2, and are more formally defined by following terminology [36]:

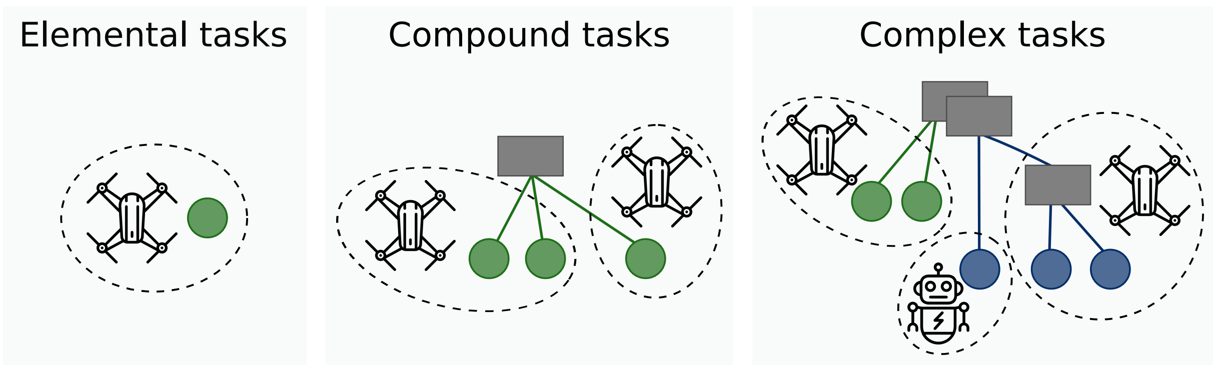


Figure 2.2: Classification of tasks in terms of their complexity.

- *Decomposition and Decomposability:* A composite task t is decomposable if it can be represented as a set of subtasks σ_t for which the completion of a given combination of subtasks in σ_t is the completion of t. The combination of subtasks that satisfy t can be represented by a set of relations ρ, which may contain constraints between subtasks or rules about which or how many subtasks are required. The pair (σ_t, ρ_t) is also called a decomposition of t. The term decomposition can also refer to the process of decomposing a task.
- *Multiple Decomposability:* A task t is multiply decomposable if there is more than one possible decomposition of t.
- *Elemental Task:* An elemental (or atomic) task is a task that is not decomposable.
- *Compound Task:* A compound task t is a composite task that can be decomposed into a set of elemental and/or compound subtasks, subject to the condition that there is exactly one fixed complete decomposition for t (i.e., a compound task must not have multiple decomposable tasks in any decomposition step).
- *Complex Task:* A complex task is a multiply decomposable composite task for which there is at least one decomposition that can be performed by the given multi-robot system. Each subtask in the decomposition of a complex task can be simple, compound, or complex.

Problem taxonomy

Over the years, there have been several taxonomies for mission planning problems. Formerly, the most widely used was by Gerkey and Matarić [37], in which problems are classified along three axes. The first axis, single-task robots (ST) versus multi-task robots (MT), distinguishes between problems where each robot can only perform one task at a time and problems where some robots can perform multiple tasks simultaneously. The second axis, single-robot tasks (SR) versus multi-robot tasks (MR), discerns problems where each task must be performed by exactly one robot and problems where some tasks may require multiple robots. On the third axis, instantaneous assignment (IA) versus time-extended assignment (TA), distinguishes problems that deal with instantaneous allocation of tasks and problems that consider both current and future allocations. In the latter case, each robot is assigned several tasks scheduled to be performed according to a certain schedule. An illustration of the taxonomy representing the problem complexity is given in Figure 2.3.

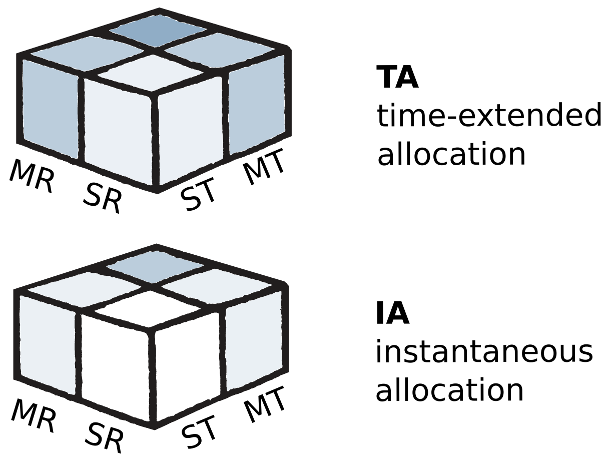
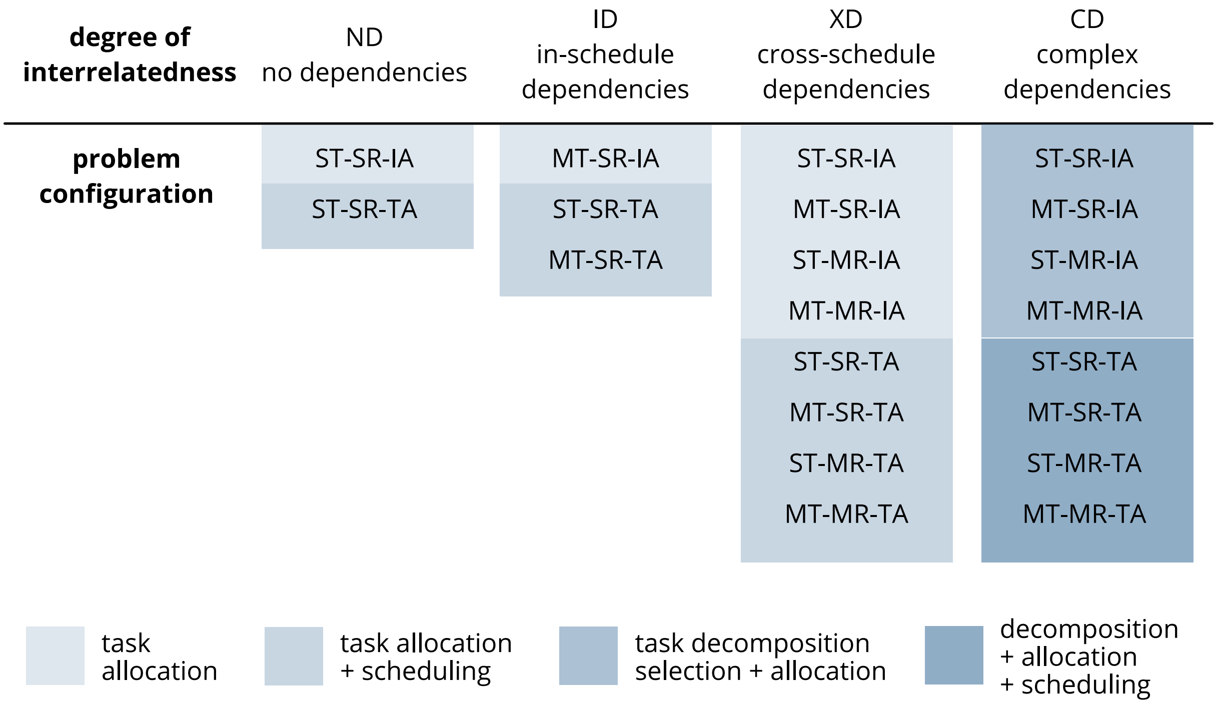


Figure 2.3: An illustration of an early three-axis classification of mission planning problems by Gerkey and Matarić [37]. Classes are color coded, with darker colors representing more complex problems.

Although it describes the potential problems very well, the taxonomy lacked an important aspect of mission planning, namely the complexity of the relationships between tasks. Korsah *et al.* [38] build upon the previous problem classification and add another axis that regards the task structure itself. Here, problems are divided into four groups:

– No Dependencies (ND) - the effective utility of a robot for a task does not depend on other tasks or robots in the system.
– In-Schedule Dependencies (ID) - the effective utility of a robot for a task depends on what other tasks that robot is performing. There may be additional constraints between the tasks in its schedule.
– Cross-Schedule Dependencies (XD) - a robot's effective utility for a task depends not only on its own schedule, but also on the schedules of the other robots in the system. For this class, the allowable dependencies are "simple" dependencies, where task decomposition can be optimally specified before task assignment. Constraints may exist between the schedules of the different robots.
– Complex Dependencies (CD) - the effective utility of a robot for a task depends on the schedules of the other robots in the system, in a way determined by the chosen task decomposition. This class of problems entails dependencies between the schedules of *complex* tasks. Therefore, the optimal task decomposition cannot be

decided before task allocation, but must be determined simultaneously.



degree of interrelatedness	ND no dependencies	ID in-schedule dependencies	XD cross-schedule dependencies	CD complex dependencies
problem configuration	ST-SR-IA	MT-SR-IA	ST-SR-IA	ST-SR-IA
	ST-SR-TA	ST-SR-TA	MT-SR-IA	MT-SR-IA
		MT-SR-TA	ST-MR-IA	ST-MR-IA
			MT-MR-IA	MT-MR-IA
			ST-SR-TA	ST-SR-TA
			MT-SR-TA	MT-SR-TA
			ST-MR-TA	ST-MR-TA
			MT-MR-TA	MT-MR-TA

task allocation
task allocation + scheduling
task decomposition selection + allocation
decomposition + allocation + scheduling

Figure 2.4: An illustration of Korsah's taxonomy of mission planning problems [38]. The different classes refer to the problems they involve (task decomposition selection, task allocation, scheduling), and are color-coded according to difficulty (more difficult problems are marked darker).

This classification can be directly related to different subproblems of mission planning (task decomposition selection, task allocation, scheduling) that need to be solved for each variety. In Figure 2.4 is an illustration of all possible mission problem classes according to Korsah [38]. Different problem difficulties are marked with lighter (easier) and darker (more difficult) shades of blue. We see that problems of class IA generally require only task allocation, since actions are immediately assigned to the robots and no scheduling is required. Similarly, problems of class TA always require a scheduling procedure, since the current and subsequent tasks are known at the beginning of the mission. Furthermore, problems of complexity CD are always accompanied by a task decomposition procedure, since there are several different ways to accomplish the mission.

It is important to point out another taxonomy, that of Nunes *et al.* [39], where they propose an extension of the taxonomy of Gerkey and Matarić taxonomy [37]. The novelty is the addition of temporal and ordering constraints to the time-extended assignment class (TA). The classification considers temporal constraints expressed in terms of time windows (TA:TW), and ordering constraints expressed in terms of synchronization and precedence constraints (TA:SP).

2.3 Mission Planning Approaches State of the Art

In this section, we specifically address solutions to the mission planning problems defined in the previous section. In particular, for better organization, we consider problems with cross-schedule dependencies (XD) and problems with complex dependencies (CD). Simpler problems are not considered in this survey because they are not direct counterparts of the problems studied in this thesis. We also regard only time-extended assignment variants (TA) of the problem, since simple instantaneous allocation problems are not relevant to this survey.

2.3.1 Problems with cross-schedule dependencies

Centralized solutions to mission planning problems rely on a central controller that allocates tasks to robots. The robots then simply execute the assigned plan. Since centralized architectures have all the information about the problem available, authors often opt for the exact approaches. They provide optimal solutions, but their computation time is usually impractical for realistic robotics applications, since these problems are NP-hard. A common approach is to model the problem as a variant of Mixed-Integer Linear Programming (MILP) and use commercially available solvers to obtain solutions [40, 41, 42]. The most commonly used linear problem solvers are CPLEX [43], Gurobi [44], ABACUS [45], and lp_solve [46].

Optimal solutions can be more efficiently computed using Branch-and-Bound [47] and its variants. Branch-and-Bound searches the state space of candidate solutions, represented as a tree, and uses upper and lower bounds on the optimal solution to prune the branches of the search tree whose cost is higher than the computed lower bounds. Some of the interesting applications in mission planning include the ones proposed in [14, 48]. In [14], the XD[ST-MR-TA] class problem is modeled as Dial-a-Ride Problem (DARP), a variant of Vehicle Routing Problem (VRP) with pickup and delivery. To solve the problem, the authors use a centralized bounded optimal Branch-and-Price algorithm (a variant of Branch-and-Bound). Although optimal, the method is computationally expensive and lacks reactivity in dynamic environments.

Since the mission planning problem is intractable for larger numbers of robots and tasks with intricate dependencies, the focus often shifts to approximations and heuristic solution methods. For example, in [12], a MILP solver is used in conjunction with a heuristic task sequencer to solve task allocation and scheduling problems more efficiently. Although the algorithm is incomplete, the authors empirically show that the algorithm produces schedules within 10% of the optimum for real-world structured problems. The proposed approach can solve problems with up to 100 agents and 1000 subtasks in $\sim 120s$.

To further increase computational efficiency, metaheuristic approaches [49] are often used. Metaheuristics are algorithmic templates that use various heuristic techniques to find approximate solutions to difficult combinatorial problems. During the search process, they allow the method to avoid getting stuck in local optima by admitting lower quality solutions. Being stochastic, they do not provide guarantees on the solutions. However, their computational speed and ability to search in a vast solution space often compensate for this drawback, and the solutions they produce are often suboptimal. Some of the common approaches in mission planning problems include application of swarm intelligence algorithms [50, 51], evolutionary algorithms [52], Iterated Local Search (ILS) [53].

The *distributed solutions* for mission planning differ greatly in their approaches. Unlike the centralized solutions, here the control is distributed among different computing nodes. Therefore, these systems are more robust, but have the disadvantage that they are usually suboptimal compared to centralized solutions and cause additional communication overhead to the system.

Some of the best known distributed solutions to the multi-robot mission planning problem are auction- and market-based approaches. They usually solve the task allocation problem, where robots use bidding mechanisms for simple tasks that they assign to each other. Sequential auction algorithms produce suboptimal solutions in a very short computation time [54]. This, together with the ease of implementation, fast execution, and easy extension to dynamic scenarios, makes auction-based approaches an attractive solution [55, 56].

In [55], the authors propose an auction-based method for a team of robots to allocate and execute tasks with temporal (time window) and precedence constraints. The robots use a priority-based, iterated, sequential single-item auction algorithm to allocate tasks among themselves. An important innovation is the decoupling of precedence from temporal constraints and the separate treatment of the two. Using simulations and experimental results, the authors have shown that this method results in schedules whose distances are close to the optimum.

Although distributed, auction-based algorithms require constant communication with a central auctioneer node to share bids and results. To alleviate this issue, solutions that use consensus algorithms for bid resolution have been suggested. A pioneering work on this area is the Consensus Based Bundle Algorithm (CBBA) [57]. Authors in [58] extended the basic algorithm to incorporate temporal constraints. However, the objective function remains rather simple, as it only accounts for a number of executed tasks.

2.3.2 Problems with complex dependencies

All of the above approaches have not addressed the problem class CD that involves task allocation and scheduling and task decomposition selection. The problems of this complexity are even more difficult to solve, and often the application scenarios do not require tasks of this type. However, there are several attempts to solve the CD class problem, and these usually focus on a single application scenario.

In [59], the authors present a multi-robot task and motion planner for multiple decomposable tasks with sequentially dependent subtasks. The planner is tailored to a case where tasks can be divided into subtasks that must be completed in a predefined order. The method is evaluated on a transportation task in a lake environment where the land and water robots can exchange a payload at dock locations. Each task in this example had three possible decompositions.

The work of Jones *et al.* [60] describes a CD[ST-MR-TA] problem with rather complex precedence and simultaneity constraints. The time-critical coordination problem is illustrated by an emergency response domain in which a group of fire trucks attempt to fight a series of fires spread across a city. The disaster has also blocked many city streets with impassable debris that can be removed by bulldozer robots. A coordination solution must not only assign tasks, but also determine which routes fire trucks should take given the assigned priority conditions within the routes and which bulldozers should be tasked with clearing debris along those routes. The proposed solution involves a market-based approach with combinatorial bidding and tiered auctions where agents can submit sub-bids to negotiate other agents' participation in their plans. The approach also involves resolving conflicts between schedules. There is no optimality guarantee in this approach, since it is designed for problem sizes larger than those that can be solved optimally.

In [4], the authors incorporate complex tasks into multi-robot task markets by including task tree auctions. Instead of trading contracts for simple tasks, task trees are offered in auctions. Complex tasks are specified as loosely coupled tasks connected by the logical operators *AND* and *OR*, similar to our mission specification. The method was tested with both centralized and distributed setup on a reconnaissance mission that required coverage of multiple areas. The tree auction method outperformed all single-stage auction task allocation algorithms.

Given the current state of the art (Table 2.1), we have identified the need to 1) propose a generic, mission-independent solution for task allocation and mission scheduling in the class XD; 2) build on this to propose a generic solution for problems in the class CD. Particularly in the class CD, most current solutions are tailored to specific application scenarios. Our goal is to define a problem modeling and solution framework in which any problem that adheres to the given format can be successfully solved.

Table 2.1: An overview of existing approaches to mission planning problems of classes XD and CD with time-extended allocation.

Reference	Class[37]	Complexity[38]	Organization	Optimal	Multi-objective
Koes *et al.* [40]	ST-MR-TA	XD	centralized	✓	
Ramchurn *et al.* [41]	ST-MR-TA	XD	centralized	✓	
Alighanbari *et al.* [42]	ST-SR-TA	XD	centralized	✓	
Korsah *et al.* [48]	ST-SR-TA	XD	centralized	✓	
Korsah [14]	ST-MR-TA	XD	centralized	✓	
Gombolay *et al.* [12]	ST-SR-TA	XD	centralized		✓
Wei *et al.* [50]	ST-SR-TA	XD	centralized		✓
Chen *et al.* [51]	ST-SR-TA	XD	centralized		bi-objective
Çakar *et al.* [52]	ST-SR-TA	XD	centralized		
Mitiche *et al.* [53]	ST-SR-TA	XD	centralized		
Nunes *et al.* [55]	ST-SR-TA	XD	distributed		
Godoy & Gini [58]	ST-MR-TA	XD	decentralized		
Motes *et al.* [59]	ST-SR-TA	CD	centralized		
Jones *et al.* [60]	ST-MR-TA	CD	distributed		
Zlot & Stentz [4]	ST-SR-TA	CD	distributed		

CHAPTER 3

Decentralized Coordination of Heterogeneous Robotic Teams

The most important prerequisite for intelligent behavior in distributed multi-agent systems is the ability of agents to independently form judgments about their actions in relation to the behavior of other agents and the overall goal of the system. To enable rational decision-making, agents must perceive their capabilities and consider the impact of their individual efforts on the mission outcome. In our work, we represent agents' capabilities, and the mission structure in a hierarchical tree form inspired by the TÆMS [61] task structure. Based on this task representation, we have developed a generic framework for mission coordination of heterogeneous robot teams, which consists of modules, communication protocols, and coordination mechanisms. It enables the intelligent control of multi-agent teams for complex missions of different domains. In this chapter, we outline the hierarchical task representation, followed by the description of the framework modules and coordination mechanisms.

3.1 Hierarchical Task Representation

The main premise of the hierarchical task representation is *task decomposition*, where large and potentially complex tasks are incrementally decomposed into simpler ones, down to the level of elementary, actionable tasks (actions). This representation provides a better overview of the mission and the relationships between tasks, and greatly simplifies mission specification. Moreover, the rich expressiveness of the mission formulation allows the definition of intricate task relations and decompositions, and thus applicability in various areas. In our work, we are guided by the general features of the TÆMS model, with adaptations to suit the developed framework.

TÆMS [61] is a framework for representing large task hierarchies, allowing the definition of simple and complex relationships between tasks and temporal constraints on their

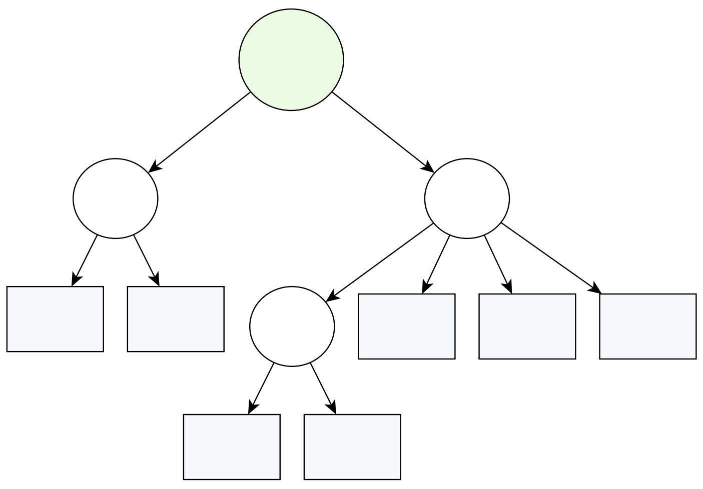


Figure 3.1: Hierarchical mission representation example. Task nodes are represented by circles and action nodes by rectangles. At the root of the tree structure is the overall mission goal (highlighted green).

execution. Models in TÆMS are abstractions of problem-solving processes decomposed into simple and complex tasks with potentially complex relations, omitting the specifics of task execution. The graphical representation of this task model is in the form of a tree (Figure 3.1), with the root task representing the overall system goal. This goal is then decomposed into several layers of subtasks, down to primitive tasks. We have taken the original concepts of the TÆMS framework and adapted some of the ideas to develop a more intuitive task representation that forms the basis for our cooperative missions. A detailed description of the hierarchical task model can be found in this section.

3.1.1 Elements of the hierarchical task model

The basis of the hierarchical task structure are *nodes*, corresponding to the basic mission elements, tasks organized in a tree-like structure, as shown in Figure 3.1. The mission tree distinguishes two types of nodes, *action* nodes that correspond to real, actionable robot behaviors, and *task* nodes that combine action nodes into a meaningful structure, as defined by the mission objective. In the original TÆMS formulation, actions are referred to as methods. In the illustration, the task nodes are represented by circles, while rectangles represent actions.

Formally, we define sets of actions and tasks as A and T, respectively. Each $a \in A$ can be performed by one or more robots. If we denote the set of robots as $R = \{1, \dots, n\}$, we can specify the set of actions robot i can perform as A_i, and the set of tasks robot i can contribute to as T_i. The root task corresponds to the mission objective. Notice that redundancy is possible, hence, in general $A_i \cap A_j$ and $T_i \cap T_j$ might not be empty sets, for $i \neq j, i, j \in R$.

The performance of actions often involves elements that are integral parts of the robot or the environment, and that the robot may require during execution, or that are a product of the action being effected. The TÆMS framework defines this concept as *resources*. They can be consumable or non-consumable and are specified with the value of an arbitrary unit of measure, depending on their type. The value of consumable resources may change as a result of action execution, as defined by special task-resource relationships. For non-consumable resources, values are affected only during action execution and are restored when the action is complete. In reality, resources model concepts such as the state of the battery charge, the occupancy of the robot tool, the illumination level, and the like. For the resource to be available, its value must be within the defined allowable limits.

Formally, a set of resources is defined as P. For each $p_i \in$ P, we define a state as $state(p_i)$. The resource availability is dictated by its limits, as

$$available(p_i) = \begin{cases} 1 & \text{if } bottom\_lim(p_i) \leq state(p_i) \leq upper\_lim(p_i) \\ 0 & otherwise. \end{cases}$$

All of the above concepts define static elements of the mission structure. The elements defined in the rest of the section combine these components and relate them to each other in complex ways as dictated by the mission objectives.

3.1.2 Task decomposition

In order to fully define task decomposition, in addition to the tree structure of the tasks, the relations between the subtasks must also be specified. These relations define the way in which the subtasks perform their respective parent task. In our model definition, the concept that specifies task decomposition is called *Quality Accumulation Function* (QAF). Similar to the TÆMS model, we distinguish several QAFs corresponding to the logical operators *{AND, OR, XOR}*. Quality is an abstract concept that depends on the problem domain and implies the contribution of a task to the achievement of the overall goal.

The function *AND* specifies the task decomposition of a task in such a way that all subtasks of a task must be executed for it to acquire quality (i.e. to be considered completed). The quality of the parent task, in this case, is computed as the *sum* of all subtask qualities. Specifying the function *OR* leads to a task decomposition where the quality of a composite task is computed as the *sum* of all subtask qualities. Any solution in which a subset of the subtasks is performed renders the parent task completed. Finally, by the definition of the *XOR* function, the task structure requires the execution of *exactly one* subtask. Thus, the quality of the parent task is equal to the quality of the executed task.

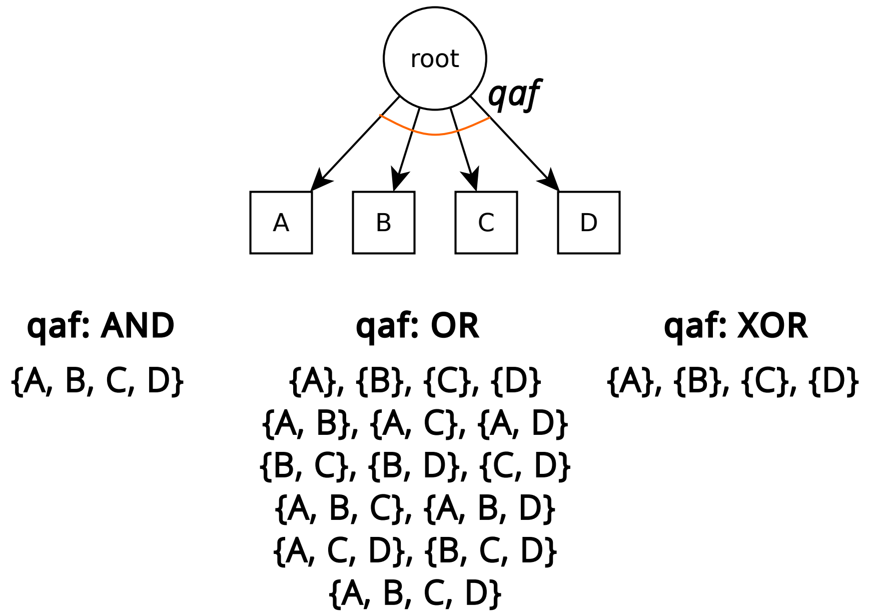


Figure 3.2: Task decomposition functions (QAFs) of the hierarchical task model. For each function, AND, OR, and XOR, a set of possible decompositions for the task is given.

All three functions given are illustrated by an example of a simple task decomposition consisting of four subtasks, as shown in Figure 3.2. For the case of the AND and XOR operators, the number of possible decompositions is small, with AND defining only one possible option and XOR defining n different decompositions (n is the number of subtasks). For the case of the OR operator, the number of ways in which a task can be executed is equal to $2^n - 1$. In this case, a rapid combinatorial explosion may occur, and scheduling algorithms must take this into account and carefully mitigate its effects.

3.1.3 Task relations

Task relations enrich the task structure by further specifying effects of task execution on other tasks and resources. In general, they can be considered as *hard* or *soft* relations. Hard relations between tasks model strong task dependencies that must be respected in order to accomplish the mission. Soft relations only exhibit effects on the quality of task execution, which can be either increased or decreased by relation activation.

The most prevalent task relations that directly influence the task ordering are *precedence constraints*. They specify the pairwise order of tasks or actions in the task structure. Formally, we define the precedence constraints as $prec(a, b), a, b \in T \cup A$ for two tasks a and b. This relation defines that task a needs to be executed before the start of the task b. By nature, they are hard constraints which need to be met for the schedule to be valid for the given task structure. In the original TÆMS representation, the precedence constraits are modelled by *enables* and *disables* interrelationships [61].

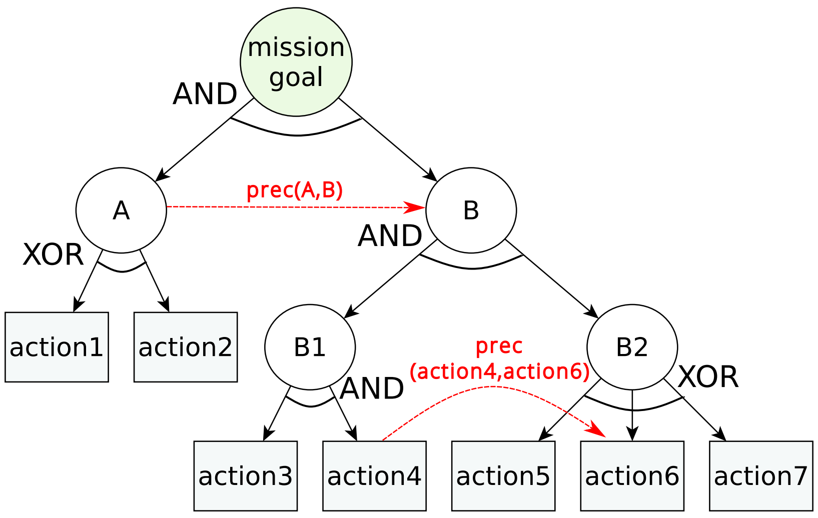


Figure 3.3: Example of a mission tree with defined precedence constraints and QAFs. Precedence constraints are specified with red dashed arrows, and QAFs are given next to each task node.

An example of a task structure with defined precedence constraints and QAF is shown in Figure 3.3. There we can observe two cases of precedence constraints, namely task-task and action-action. The latter case is much simpler, since the relation only affects the two actions directly. If an element of the precedence relation is a task, all its subtasks are affected by this relation. In this example, all subtasks of task A must precede all subtasks of B in the final solution. As mentioned before, it is also possible to define task-action and action-task precedence.

Other relations defined by the TÆMS model are the relations *facilitates* and *hinders*, both soft constraints. They define how the performance of one task affects another task when executed in the specified order. For example, if certain preparatory actions can increase the quality of a task, they can be modeled as a separate task. Then we can model this synergistic effect between the two with the relation facilitates. If the opposite is true, that is, if the execution of one action decreases the quality of a subsequent action, this can be modeled by a hindering relation. These constraints are called soft because they do not render the solution infeasible if they are broken. However, their effects directly penalize or reward the goodness of the solution obtained. Similarly to the precedence, these constraints are defined as $facilitates(a, b), a, b \in T \cup A$, and $hinders(a, b), a, b \in T \cup A$. These relations are currently not needed and used in our model, however, they can be easily integrated as a future work and are therefore represented in the thesis.

Another set of constraints relates to the concept of resources mentioned earlier, where the relations *consumes* and *produces* define how the execution of certain tasks affects the state of the respective resource. These constraints are defined as task-resource and action-resource relations. Similarly, there are *limits* constraints that originate from resources and restrict the execution of actions that require a particular resource. These constraints are activated as soon as the resource is not within the allowed limits. Formally, these relationships are defined as $consumes(a, p), a \in T \cup A, p \in \mathrm{P}$, $produces(a, p), a \in T \cup A, p \in \mathrm{P}$, and $limits(p, a), p \in \mathrm{P}, a \in A$.

After all the building blocks of the task structure and their intricate relationships have been defined, the next step is to specify how the tasks are quantified and evaluated.

3.1.4 Task evaluation

Each task is quantitatively described in three dimensions: quality, duration, and cost. As stated previously, quality is an abstract concept that defines the contribution of a task to the achievement of the overall goal. Duration represents the time required to complete a particular task, and cost is the cost incurred to perform the task (which can be energy expenditure, financial cost, resources consumed, etc.).

To evaluate the tasks, each $a \in A$ is assigned a triple $(k_a(i), d_a(i), c_a(i))$, where $k_a(i)$ is the action quality, $d_a(i)$ is the duration, and $c_a(i)$ is the action cost when performed by robot $i \in R$. The action quality is determined a-priori by the system designer. Each robot estimates the duration and cost of a future action based on the current state of the system and their capabilities. The outcome of each task $t \in T$, (k_t, d_t, c_t), is determined using the quality accumulation function $Q : T \to \mathbb{R}^3$, which describes how subtasks contribute to the quality of a higher-level task. In general, the function Q can have any user-defined form. In this work, we model the function Q with respect to the previously defined quality accumulation functions (QAFs).

3.2 Development of Coordination Framework for MRS

The hierarchical mission representation described earlier is the backbone of the mission planning and coordination framework used in this thesis. Its main premises are domain independence and a generalized approach to multi-robot coordination. In developing the framework, we took inspiration from *Generalized Partial Global Planning* (GPGP) [62], which is often used in synergy with the TÆMS task structure. GPGP serves as a roadmap for distributed coordinated control of multi-agent systems by providing general guidelines for the formulation of coordination mechanisms and module structure.

In this section, we outline the coordination framework for multi-robot teams. The development of the system starts with the initial adaptation of the GPGP framework, which later evolves into a new distributed mission coordination framework. In this section, we first describe the original GPGP framework (Subsection 3.2.1) and identify its main issues. Then, in Subsection 3.2.3, we build upon the basic concepts of GPGP and address its weaknesses of decoupled task allocation and scheduling processes and overly complicated coordination mechanisms due to local views of the mission. The new framework maintains all the positive sides of GPGP framework, with improved robustness, speed and optimality.

3.2.1 GPGP framework infrastructure

GPGP is intended to be a highly modular decentralized coordination framework for multi-agent missions defined in terms of TÆMS trees. The framework defines various functional modules, as well as coordination protocols that enable cooperative mission execution. Coming from a computer science background that considers generic intelligent systems, agents in this framework represent entities that participate in the coordination process. In robotic systems, agents refer to individual robots.

Figure 3.4 shows a typical structure of an agent with GPGP infrastructure. The core of any GPGP agent is the TÆMS structure with specified mission models that represent its capabilities to perform various tasks. The component that is tightly coupled with the TÆMS structure is *Task Assessor*. This domain-specific module is aware of the capabilities of the robot and the specifics of how it performs various actions. Its task is to evaluate all actions $a \in A$ of a given TÆMS structure at the beginning of each mission and compute their $(k_a(i), d_a(i), c_a(i))$ for $i \in R$ with respect to the current state of the system. It is a rather simple module to implement for each TÆMS structure in the robot's knowledge base.

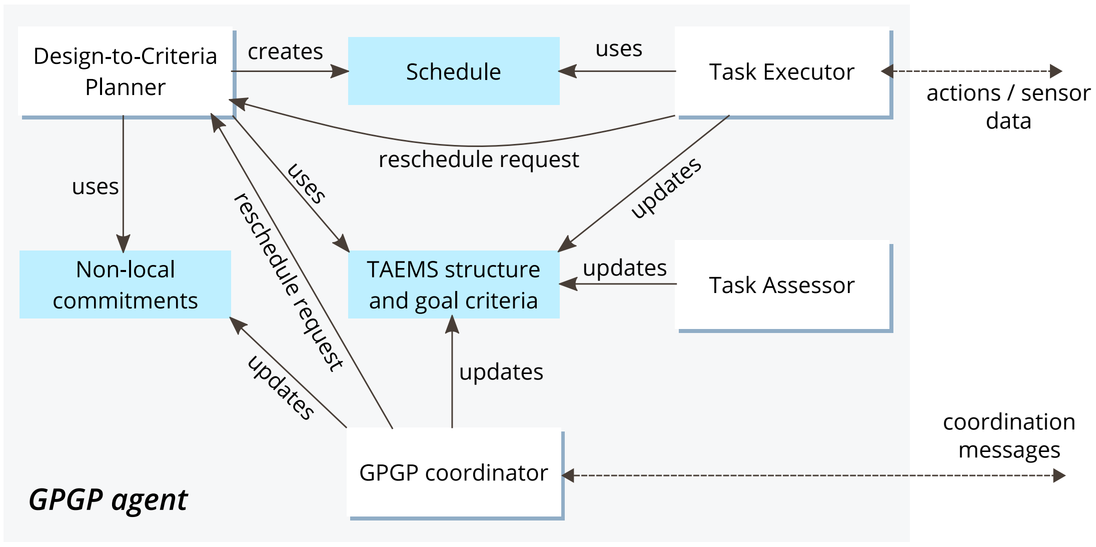


Figure 3.4: Structure of an agent with GPGP infrastructure. The modules are depicted in white rectangles, while blue rectangles represent different data structures.

Design-To-Criteria planner *(DTC)* [63] is a component that generates a robot's schedule based on the TÆMS structure and the temporal constraints dictated by commitments.

Commitments represent a contractual concept between two robots that guarantees synchronous execution of constrained tasks. Each robot plans its schedule locally, taking into account deadlines imposed by commitments to other robots (originating from constraints defined in Section 3.1). For large-scale systems, the DTC Planner handles a very complex scheduling problem by using various heuristic methods and approximations to construct a schedule that satisfies certain user-defined criteria. Although criteria can be richly specified in terms of five multidimensional objective functions that are merged into a common measure [63], we only optimize in the basic (k, d, c) space. The tasks defined in our missions do not contain *uncertainty* about the quality, duration, and cost of the tasks supported by the TÆMS framework.

The schedule is used directly by *Task Executor*, a module that initiates the timely execution of actions and ensures that each action is properly executed before advancing the schedule. It ensures that the hard relations between tasks are maintained, even though the execution times may differ from the estimated ones. If execution times change by a significant amount from those originally planned, the scheduling process is restarted.

Finally, the *GPGP Coordinator* is a component that takes care of the schedule with respect to other robots, as it ensures that the actions of all robots are coordinated. It uses coordination mechanisms proposed in [64] and further refined and implemented in our work, which are described in the next paragraphs, and stated as follows:

- updating non-local viewpoints,
- communicating results,
- handling simple redundancy,
- handling hard coordination relations,
- handling soft coordination relations.

Starting from this agent structure, we now describe the planning and coordination process itself. Interactions between robots ensure coordinated execution of tasks. Planning and coordination are performed in a decentralized manner in several steps, implementing the coordination mechanisms mentioned above. In this process, the problems of task decomposition selection, task allocation, and scheduling are performed sequentially, which we later identified as a weakness of this framework. In developing our framework, we have evolved the original sequential design to provide an integrated process for task allocation and scheduling, which is explained in Subsection 3.2.3.

3.2.2 GPGP scheduling and coordination procedure

The most prominent feature of the GPGP framework is the unawareness of each robot about the capabilities of the other robots, which manifests itself in a local view of the task structure. The rationale for such an approach is to increase the modularity of the system,

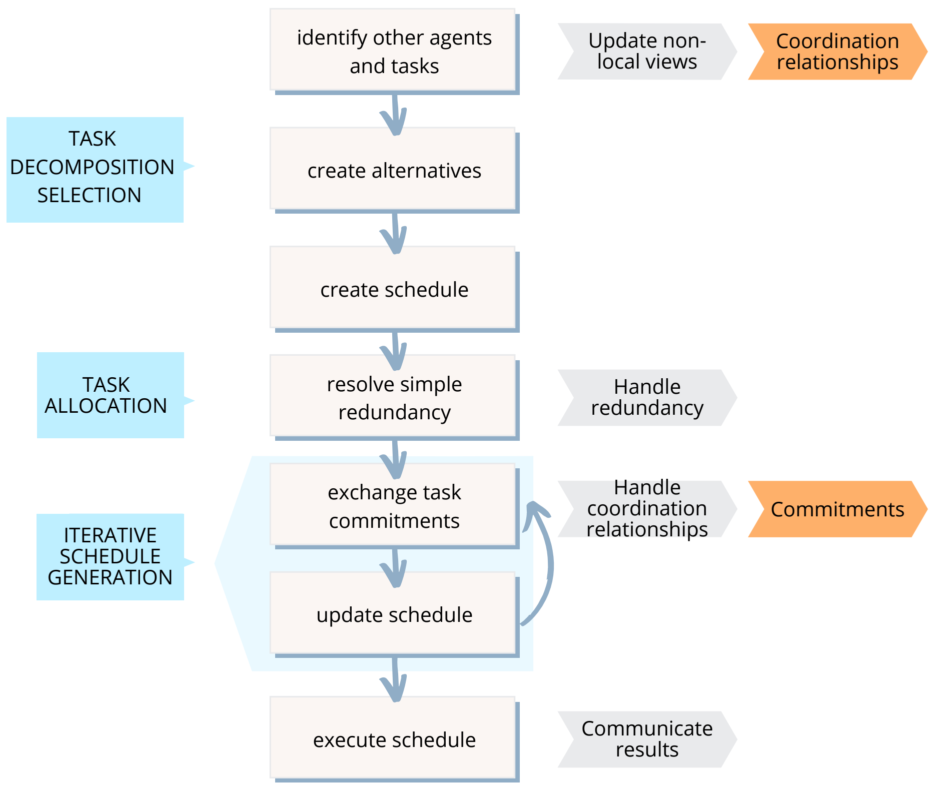

identify other agents and tasks
create alternatives
create schedule
resolve simple redundancy
exchange task commitments
update schedule
execute schedule
TASK DECOMPOSITION SELECTION
TASK ALLOCATION
ITERATIVE SCHEDULE GENERATION
Update non-local views
Coordination relationships
Handle redundancy
Handle coordination relationships
Commitments
Communicate results

Figure 3.5: The diagram of the GPGP process. The different mechanisms present in each stage are marked on the right side of the graph.

since robots have no prior knowledge about other participating robots. Therefore, a local view of the mission is created for each robot type that can participate in the mission execution. The nodes contained in it are actions that robots can perform and tasks that they can contribute to. For robot $i \in R$, the nodes contained in the mission structure are those representing tasks T_i and actions A_i, as previously defined in Section 3.1. The mission coordination process ensures the mutual identification of the robots involved and the updating of the local views with minimal information requirements.

Figure 3.5 shows the diagram of the whole GPGP coordination process, with the different mechanisms present in each stage marked on the right side of the graph. The first stage of the coordination procedure is the process of identifying other robots and their tasks. In this stage, the GPGP mechanism *updating non-local views* is issued. Here, the robots exchange their locally estimated task and action outcomes. Coordination relationsips can arise from constraints on tasks (hard and soft relations) or from *parent-child* relations between tasks.

The second step is *generation of task alternatives*, where the best task decomposition for mission execution is selected. We define a *task alternative* $alt(t)$, $alt(t) \subseteq A$, $t \in T$ as an unordered set of all actions whose execution leads to the completion of task t. The sizes (cardinal numbers) of the task alternative sets for different tasks in the mission plan depend on the structure of the mission tree and the relationships between the nodes. For highly constrained missions, the cardinal numbers are generally small (i.e., $O(1)$). On the other hand, for missions without any node relationships, the combinatorial explosion can lead to a factorial size complexity of the task alternative generation procedure.

Given the decomposition determined in the previous step, the initial schedules for each robot are constructed. Based on these schedules, the *handle simple redundancy* procedure assigns a single robot to each task, thus solving the task allocation problem. Redundant tasks for a robot $i \in R$ are identified as $\{t \in (A_i \cup T_i) \cap (A_j \cup T_j), j \in R, j \neq i\}$. Redundancy resolution is followed by the procedure of *iterative schedule construction*, which determines the order of actions for each robot, taking into account the temporal constraints between them. During the schedule construction, the robots consider all the exchanged commitments and plan a schedule that respects them. If some of the commitments are broken by the schedule update procedure, committing and scheduling is repeated until all the constraints are satisfied. Once all the robot schedules align, the task execution module receives the schedule and starts executing it. After the execution of each task, the results of its performance are communicated to all robots that rely on it.

After the brief overview of the overall coordination process, a more detailed look at the main coordination mechanisms follows.

Update of non-local views

Since the knowledge of each robot is limited to a local mission representation (based on robot abilities and defined by system designer), it needs to collect necessary information from other robots. To minimize the amount of exchanged information, robots share only estimated execution outcomes given as (k, d, c), thus encapsulating the details behind task execution. It is important to identify the minimal set of tasks to be exchanged in order to ensure the most complete possible task structure update, without sharing unnecessary information. The coordination relationships stem from the two main sources, as follows.

Coordination relationships are detected between robots i and j for related actions/nodes:

(i) constrained tasks $\forall t_a \in (T_i \cup A_i), \forall t_b \in (T_j \cup A_j)$ s.t. $\exists constr(t_a, t_b) \vee constr(t_b, t_a)$, for $constr \in \{prec, facilitates, hinders\}$

(ii) child tasks $\forall t_a \in T_i, \forall t_b \in (T_j \cup A_j)$ s.t. $\exists pc(t_a, t_b)$, where pc denotes the parent-child relation

The example of the global and local mission views and the identified coordination relationships (CRs) is shown in Figure 3.6. The CRs $(task_1, A)$ and $(task_1, B)$ are identified based on parent-child relationships between the tasks of the two robots $robot_1$ and $robot_2$. The relationship $(task_1, C)$ is derived from the precedence condition for the execution of the two tasks.

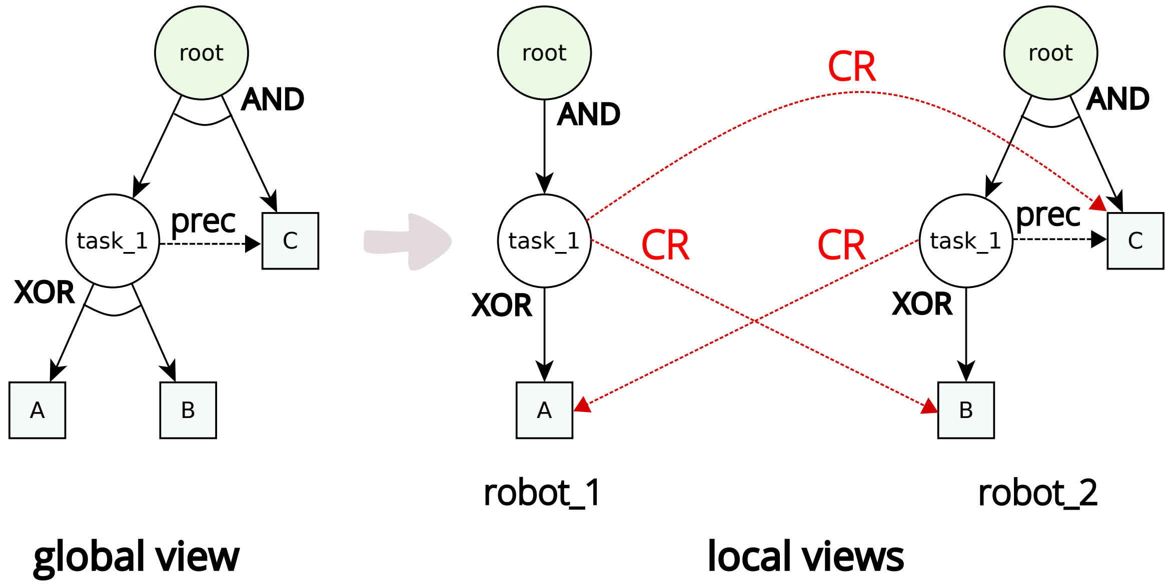


Figure 3.6: An example of global and local mission views. Two participating robots have identified coordination relationships (CR) stemming from the precedence and parent-child relations, denoted with red dashed arrows on the graph.

Determination of the best task alternatives

At the beginning of this procedure, each robot has complete information on the outcome estimates for tasks in its local mission representation. The process of task alternative generation begins at the action nodes of the local TÆMS tree and builds up recursively, finally ending at the root of the tree.

Naturally, each task has many different ways of being realized so the task alternative generation procedure uses a method of focusing the solution search by pruning the worst partial results in each step of the process, thus making the problem tractable. Furthermore, since robots at this point share the same outcome estimates, the same alternative set for common tasks is going to be obtained by each robot.

During this procedure, the robots use estimated values for quality, duration, and cost of actions, which are determined as the average of these values for all robots that can perform the action, as follows:

$$(\overline{k}_a, \overline{d}_a, \overline{c}_a) = \sum_{i \in \rho_a} \frac{(k_a(i), d_a(i), c_a(i))}{|\rho_a|}, a \in A. \tag{3.1}$$

ρ_a defines the set of robots that can perform action a, $\rho_a = \{i, i \in R, a \in A_i\}$, $\forall a \in A$. $|\rho_a|$ stands for the cardinality of the set ρ_a.

Finally, the score for each task alternative $alt(t)$ is computed based on the expected values of actions $(\overline{k}_a, \overline{d}_a, \overline{c}_a)$, and given the quality accumulation function Q and the defined tree structure. If we specify the alternative outcome values as $(k_{alt}, d_{alt}, c_{alt})$, our simplified objective function (score) for an alternative is defined as

$$score(k_{alt}, d_{alt}, c_{alt}) = \alpha k_{alt} - \beta d_{alt} - \gamma c_{alt},\ \ \alpha, \beta, \gamma \in \mathbb{R}, \tag{3.2}$$

where $\alpha + \beta + \gamma = 1$, and they represent the importance weighting of each specific criterion. Based on these factors, the planning strategy adapts and selects the appropriate tasks to execute. With this score function, we are able to define the importance of each problem parameter (quality, duration, cost) in the task decomposition selection. As a result of this process, the robots are given a set of alternative ways to achieve the mission goal (root task).

DTC planner – iterative schedule construction

Given a root task alternative $alt(root)$, the goal of a scheduling algorithm in general is to build a schedule s_i for each robot $i \in R$, which is defined as $s_i = \{(a, a^s, a^f)\ \forall a \in S_i\}$, where S_i is the set of actions assigned to robot i and $a^s (a^f)$ are the start (finish) times of action a. Since each robot is constrained to a local view of the mission, it constructs only local schedule s_i, taking into account actions of other robots by commitments. The overall schedule s is a superposition of the individual robot schedules.

As inputs to the scheduling procedure, each robot knows a set of actions it can perform and the relationships with the tasks of other robots. The problem itself is a form of a job shop. Scheduling is done iteratively according to the Algorithm 1.

As stated in Algorithm 1, each robot first constructs an initial schedule s'_i and sends its *commitments* to other interested robots. As mentioned earlier, commitments arise from precedence constraints imposed on tasks. Formally, each commitment is defined as $comm(t_a \in T_i \cup A_i, end\_time(t_a), t_b \in T_j \cup A_j)$, $i \neq j$, $i, j \in R$ and is stored in the robot's commitment base. Robot i is thus committed to perform task t_a until time $end\_time(t_a)$, and t_b denotes the constrained task of robot j. In each next iteration, the robot adjusts its schedule according to the received commitments of other robots and its own unfulfilled commitments. The procedure is repeated until all commitments are met.

The procedure of scheduling for individual robots can be performed, for example, with a genetic algorithm, as it shall be described later. All other job shop type schedulers can also be applied to this problem.

3.2.3 GEM coordination framework

Inspired by the previously described GPGP framework, we developed our framework for multi-robot mission coordination. We identified some of the main weaknesses of the GPGP approach and updated the framework to better fit the needs and capabilities of modern

Algorithm 1: Iterative scheduling procedure for robot i.

```
input  : $alt_i$ – mission alternative of robot $i$
input  : $tree$ – mission tree
input  : $R$ – set of peer robots
output : coordinated local schedule $s_i$
Function generate_schedule($alt_i, tree, R$):
    /* generate initial schedule                                   */
    $s'_i \leftarrow$ make_schedule($alt_i$);
    forall $k \in R \cup \{i\}$ do
        $Compl_k$ = tasks completed by robot $k$ while executing $s'_k$;
    end
    /* initialize commitments                                      */
    $Comm_j \leftarrow \{\} \ \forall j \in R$;
    while $True$ do
        forall $j \in R$ do
            /* update commitments and send them to peers           */
            forall $t_a \in Compl_i,\ t_b \in Compl_j,\ prec(t_a, t_b)$ do
                $Comm_j \leftarrow Comm_j \cup \{comm(t_a, end\_time(t_a), t_b)\}$;
            end
            send $Comm_j$ to robot $j$;
        end
        /* check for the stopping criteria                         */
        if $\mid Comm_j \mid = 0,\ \forall j \in R$ AND no new commitments received then
            $s_i \leftarrow s'_i$;
            break;
        end
        /* update schedule respecting commitments                  */
        $s'_i \leftarrow$ make_schedule($alt_i, Comm$);
    end
    return ($s_i$);
```

robotic systems. Although theoretically very interesting, some of the GPGP mechanisms introduce unnecessary deviations from the optimal task scheduling solution, on the premise of reducing the availability of information to all robots in the system. The idea of keeping only local views of robot missions and then sharing the necessary information can be easily mitigated by letting all robots keep the global mission structure. Data storage and communication capabilities have evolved exponentially in modern computer systems since the GPGP was created, so a different approach is easily possible. Moreover, the DTC scheduler, which only considers local schedules and shares commitments with other robots, can be replaced by modern distributed task scheduling solutions that solve task allocation and scheduling simultaneously.

Therefore, we developed the framework *GEM*[*] (*GEneric Multirobot mission coordina-*

[*]`https://github.com/barbara0811/GEM_mission_control`

tion and planning based on hierarchical task representation). The framework is implemented in *Robot Operating System* (ROS) [65] and can be easily applied to different multi-robot teams once the specifics of each system are implemented, as defined in the framework documentation. We have retained some of the original concepts of GPGP, such as the modular organization of the architecture, which allows for easier and more intuitive development and maintenance of the system. The structure of a GEM agent is shown in Figure 3.7.

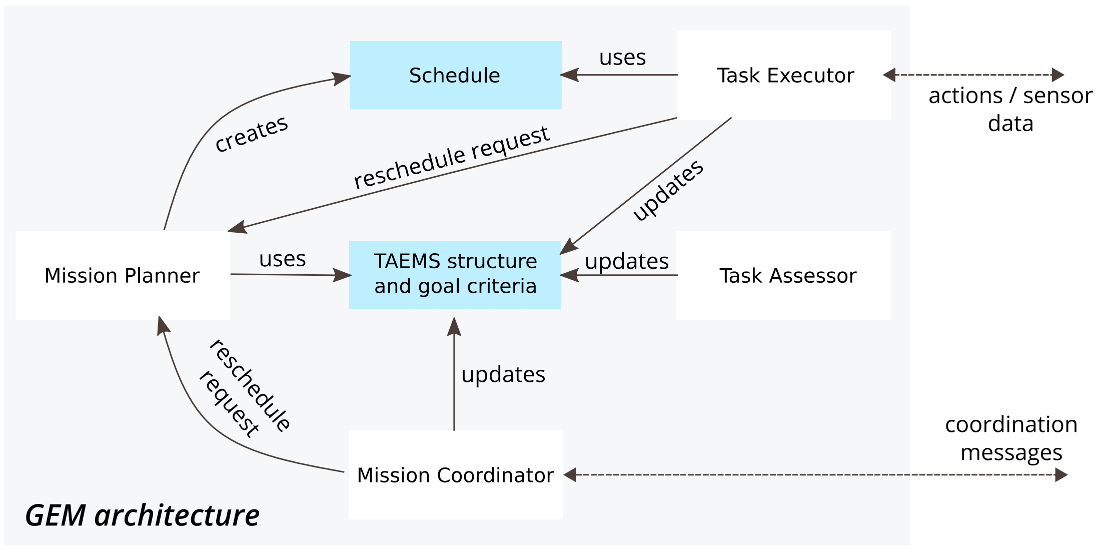


Figure 3.7: Structure of an agent with GEM infrastructure. The modules are depicted in white rectangles, while blue rectangles represent different data structures.

GEM architecture

The basic idea of separating domain-specific and generic modules is maintained in the GEM architecture. For specific applications it is necessary to implement *Task Assessor* and *Task Executor* modules. The framework provides blueprints for specific implementations in template classes that ensure seamless integration with the rest of the framework. Users must provide their specific function implementations to support the desired new functionality for different missions for the multi-robot system.

The mission-agnostic modules *Mission Planner* and *Mission Coordinator* are implemented and constant for different application scenarios. A custom implementation of the Mission Planner module is provided in the form of a distributed genetic algorithm with mimetism and knowledge sharing, as described in the following chapters. However, the user is free to replace the default planner with a custom implementation. Both modules are designed with a fairly simple interface to the rest of the system. The Mission Coordinator communicates with other robots in the system and coordinates the entire planning and execution process. This module is simpler than its GPGP counterpart, as some GPGP mechanisms have been removed from this framework.

GEM coordination process

The GEM coordination process diagram is shown in Figure 3.8. The problems introduced in the GPGP framework regarding local views of the mission are mitigated by making the global mission structures available to all robots. This modification reduces the need for many complex coordination mechanisms and facilitates the optimization procedure to find better solutions. Each robot evaluates its missions, and in the mutual identification step, it shares this information with other participating robots. Similar to GPGP, the alternative ways of completing tasks are generated from the mission tree, and several best ones are selected for scheduling. This enables the overall procedure to consider several possible task decompositions and find better solutions to the given mission planning problem. The important difference and advantage to the GPGP is that the task allocation and scheduling procedures can be integrated into a single algorithm. This is a major advantage since the assignments affect the final schedules in complex ways. The ability to optimize both simultaneously can ultimately lead to better overall solutions.

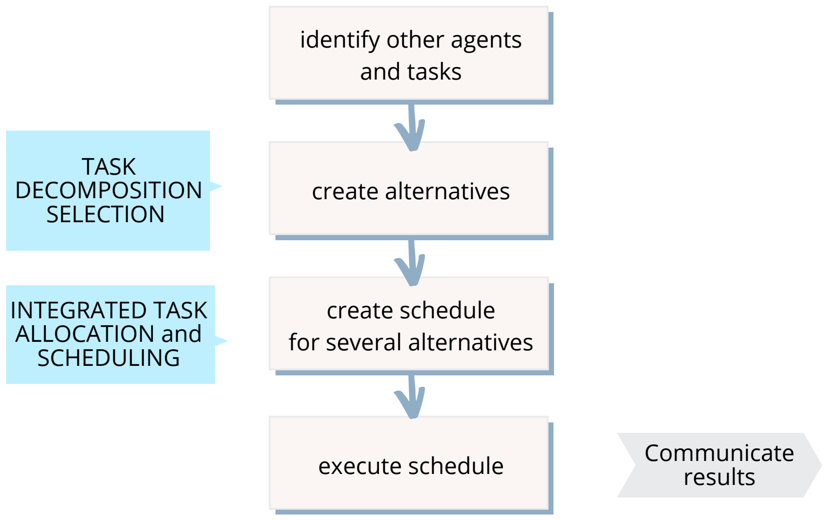

identify other agents and tasks
TASK DECOMPOSITION SELECTION
create alternatives
INTEGRATED TASK ALLOCATION and SCHEDULING
create schedule for several alternatives
execute schedule
Communicate results

Figure 3.8: The diagram of the GEM process. Different subproblems of the mission planning are marked on the left side of the figure.

This generic framework provides an excellent base for the development of specific optimization procedures for multi-robot mission planning, as follows in the next chapters.

CHAPTER 4

Definition and Modeling of Task Planning Problems

One of the contributions (Contribution 3) of this thesis is the development of a solution for time-extended task planning of heterogeneous multi-robot systems for problems with tightly coupled *complex tasks* (class CD[ST-MR-TA], as defined in the taxonomy in Section 2.2). Planning problems of this type involve cross-schedule dependencies, including precedence and temporal constraints. Here, the effective utility of an robot-task pairing depends heavily on the schedules of the other robots in the system and their respective task decompositions. The CD problems are modeled using task hierarchies described in Section 3.1, and their solutions are given later in Section 5.2.

The tasks we model in this section fall into the simpler XD[ST-MR-TA] class of problems. This class is a sub-variant of CD problems with determined task decompositions. The XD[ST-MR-TA] tasks may require one or more single-task robots for their execution. We assume that the multi-robot tasks in the problem may consist of a fixed number of single-robot tasks connected by constraints such as precedence and synchronization. Therefore, for the purpose of modeling, we transform the problem into the class **ST-SR-TA**, and the connection between MR comes from constrains.

The novelty of our approach is based on the fact that by representing the problem as a variant of Vehicle Routing Problem (VRP), we can define a *generic model of task planning problems* that can be applied to problems from different domains of multi-robot and multi-agent systems. Thus, the proposed solution acts as a generic planner of task allocation and scheduling problems defined in terms of this model.

In this section, we consider three mathematical formulations. First, we formally define the task planning problem, followed by the mathematical definition of VRP problems, and finally, we propose a unified model that combines the two paradigms. The proposed solution for the task planning problem is based on the unified mathematical model.

Assumptions. Our problem formulation assumes tasks expressed in an appropriate form and suitably decomposed for representation as a fixed number of single-robot tasks

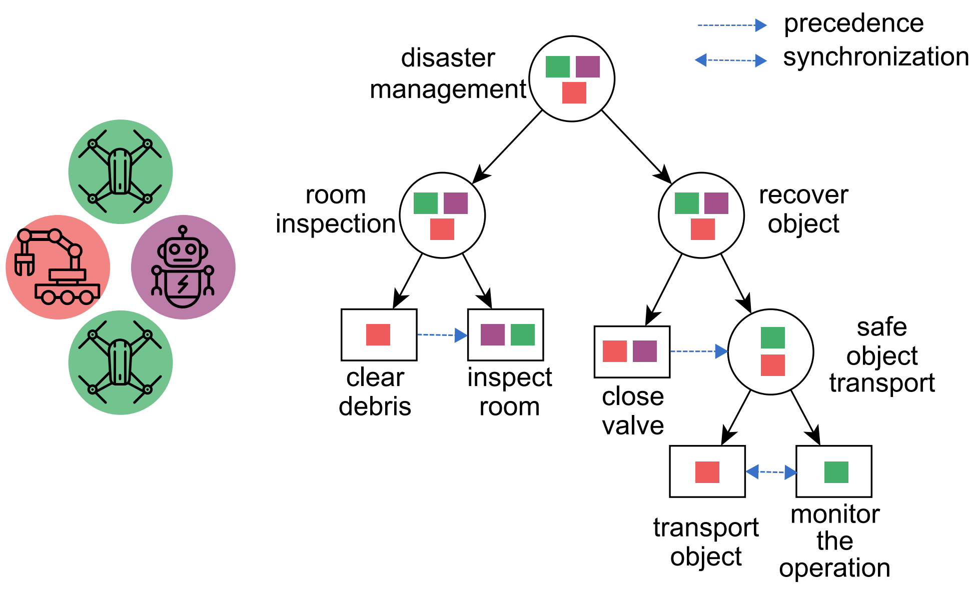


Figure 4.1: An example of a composite task with cross-schedule dependencies for disaster response mission using a heterogeneous robotic system.

related by precedence and synchronization constraints. We assume that for complex tasks, task decomposition is performed a priori. Therefore, we exclude scenarios where there are complex tasks with multiple possible decompositions (CD class) and model at most tasks with cross-schedule dependencies (XD).

4.1 Problem Statement

We consider a problem where a team of heterogeneous robots is available to perform various tasks. Any *composite task* that can be performed by a single robot or collaboratively by multiple robots can be decomposed into multiple simpler tasks. Its subtasks can, in turn, be composite tasks or simple single-robot tasks (*actions*), which can be related by precedence and synchronization constraints. They can be further constrained in time by introducing time windows. At the lowest level of task abstraction, each single-robot task ultimately consists of a set of actions that can be executed in a particular way to complete the task. An illustrative example of a representative problem is shown in Figure 4.1. In it, a team of four robots is used to solve a complex problem with cross-schedule dependencies. Since these are single-robot tasks, all actions in the task structure must be assigned to only one capable agent through a task assignment procedure. Multi-robot tasks are implemented by organizing simple tasks and enforcing synchronization constraints.

Definition 1 Actions and tasks. *In our model, we denote the set of actions as A and the set of tasks as T. Each $a \in A$ can be executed by one or more robots. If we denote the set of robots as $R = \{1, \ldots, m\}$, we can specify the set of actions that robot i can perform as*

A_i and the set of tasks that robot i can contribute to as T_i.

Note that redundancy is possible, so in general $A_i \cap A_j$ and $T_i \cap T_j$ may not be empty sets, for $i \neq j, i, j \in R$. This notion further complicates the task of the planning procedure, but greatly increases the overall robustness of the system. For example, in a search and rescue scenario, we can use different capabilities of different robots to accomplish the mission more effectively. For instance, robots with greater agility and space coverage capabilities can perform surveillance and inspection tasks. In contrast, slower and more robust agents can administer debris removal or transport of objects with lower urgency. In an illustrative example in Figure 4.1, we imagined a heterogeneous team consisting of a mobile robot with one robotic arm on board, two agile UAVs, and a humanoid robot with two manipulator arms. The slow mobile robot is predestined for the tasks of debris removal and object transportation, while the UAVs serve as inspection and surveillance devices. The two-armed humanoid robot can perform delicate manipulation as well as reconnaissance tasks.

In our model, we recognize a variety of constraints that affect the behavior of the system in mission execution. First, we consider precedence constraints, which are essential to represent the real needs for executing a mission in a certain order. By defining these constraints in the mission, we can easily specify the desired execution order for specific subsets of tasks. Figure 4.2 shows two different types of precedence constraints, the intra-schedule and the cross-schedule variants. In the first variant, both related tasks belong to the schedule of the same robot, while the second variant combines tasks of different robots.

Definition 2 Precedence constraints. *If the action $a \in A$ must be completed before the action $b \in A$ starts, we can specify a constraint between the two as $prec(a, b)$. This constraint forces $a^f < b^s$, where a^f and b^s indicate the times when the action a finishes and b starts.*

Precedence constraints can also be defined for composite tasks by relating all actions of one task to the set of all actions of the other task. In the example of the disaster response scenario presented earlier (Figure 4.1), such relations could be useful for tasks where safety is a concern. An example of such a constraint is *prec(close valve, safe object transport)*, where the primary task is to close the valve after the disaster has occurred, and only then enter the room with robots to transport objects to safety. The same restriction applies to the tasks *(clear debris, inspect room)*, since some debris lying on the way must be removed to enter the room.

Similarly, we consider an even stricter restriction on robots' schedules, which is *synchronization* constraint. It requires two tasks to execute at the same time, i.e., the start times

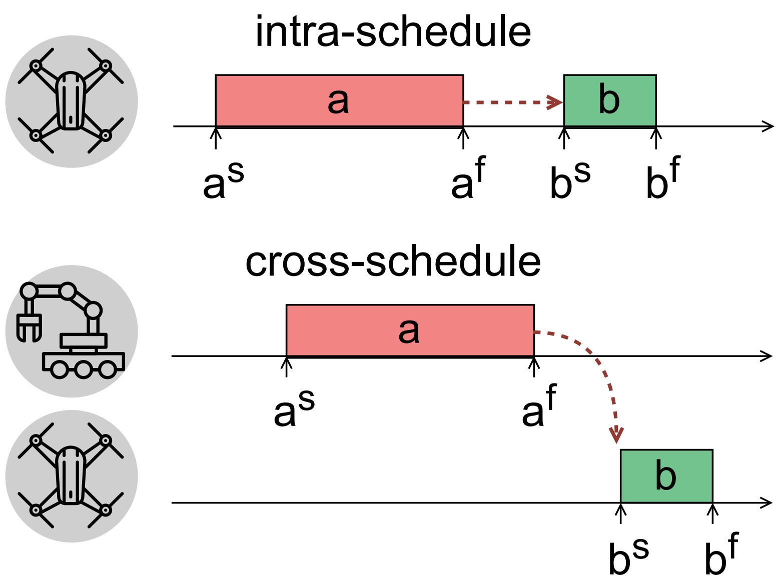


Figure 4.2: Illustration of the precedence constraint. There are two types of precedence constraints: intra-schedule (regarding tasks of a single robot) and cross-schedule (regarding multiple robots).

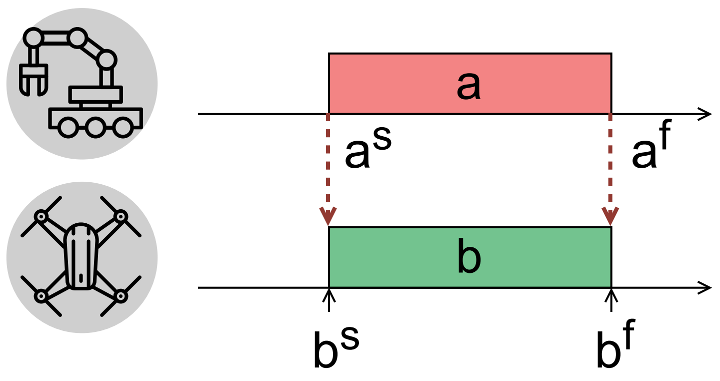


Figure 4.3: Illustration of the synchronization constraint. This relation enforces the concurrent execution of two tasks. Therefore, the start and end times of the tasks in the robots' schedules must be aligned.

and end times of their execution must be the same. Since we are dealing with single-task robots, this constraint inherently implies problems with inter-schedule (or cross-schedule) constraints, as illustrated in Figure 4.3.

Definition 3 Synchronization constraints. *Formally, we define the synchronization constraint for two actions* $a, b \in A$ *as* $sync(a, b)$. *This constraint is realized in two restrictions* $a^s = b^s$ *and* $a^f = b^f$, *where* (a^s, b^s) *and* (a^f, b^f) *denote the instances at which the actions* a *and* b *start and finish, respectively.*

As mentioned earlier, we introduce this constraint mainly for modeling multi-robot tasks. The idea is to relax a problem into an instance of single-robot tasks so that they fit more directly into the VRP model, and then introduce constraints on the model to achieve joint task execution. This type of modeling can facilitate, for example, disaster response problems where a robot must oversee the execution of a high-risk operation. In our example, we have designated a UAV to monitor the transport of a sensitive object performed by a UGV. The UAV has an innate field of view advantage and can significantly assist the transport operation. We can define such a constraint as *sync(transport object, monitor the operation)*.

The next constraint we consider in our model is involving time windows to the schedules. This type of requirement can be specified when a particular task must be performed within a strict time frame, as shown in Figure 4.4.

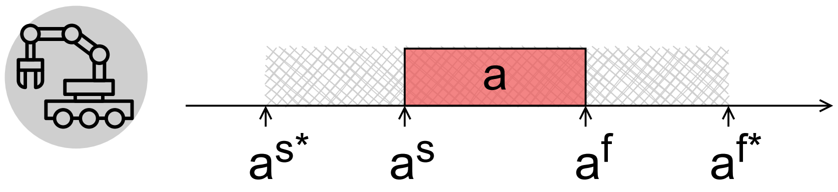


Figure 4.4: Illustration of the time window constraint. This constraint requires that tasks begin and end within a specified time window, as shown in the figure.

Definition 4 Time window constraints. *Formally, we represent the time window for action* $a \in A$ *as* $tw(a, a^{s*}, a^{f*})$, *where* a^{s*} *and* a^{f*} *stand for the earliest start time and the latest finish time of action* a, *respectively. This constraint ensures that* $a^s \geq a^{s*}$ *and* $a^f \leq a^{f*}$.

An example of a task where this limitation may be required is media coverage, where a UAV must film an object moving through the scene at specific times when the object would be in the frame.

Another naturally occurring limitation inherent in the physical system itself is the *capacity constraint*, actualized in the limited battery resources of each robot. Each action requires a certain amount of energy to perform. This amount varies for each robot and depends on its physical characteristics, along with the current position of the robot with respect to the location of the task. Therefore, another constraint that must be considered when planning task assignments is the capacity of each robot. Formally, we define the constraint as follows.

Definition 5 Capacity constraint. *For a robot $i \in R$ with capacity Q_i, we define the capacity constraint as $\sum_{a \in S_i} q_i(a)$, where S_i is the set of tasks assigned to robot i to execute, and $q_i(a)$ is the energy demand of task a for robot i.*

A solution to the described mission planning problem is a set of time-related actions (*schedule*) for all robots that does not violate the specified constraints. Formally, the schedule s_i for each robot $i \in R$ is defined as $s_i = \{(a, a^s, a^f)\ \forall a \in S_i\}$, where S_i is the set of actions assigned to the robot i and $a^s(a^f)$ are the start (finish) times of action a.

In the evaluation procedure, each single-agent task $t \in A \cup T$ is assigned a triple $(k_t(i), d_t(i), c_t(i))$, where $k_t(i)$ is the task quality, $d_t(i)$ is the duration, and $c_t(i)$ is the task cost when performed by robot i. Task quality is a subjective measure that depends on the application and is determined a-priori by the system designer. Each robot estimates the duration and cost of a future task based on the current state of the system.

The planning procedure aims to find a solution that satisfies all constraints and maximizes the global reward of the system. The objective function can be chosen arbitrarily depending on the requirements of the system. In the case of multi-objective optimization, optimal decisions must be made in the presence of tradeoffs between two or more objectives that may be in conflict. More details on our solution approach follow in Section 5.1.

To summarize, the problem features and constraints in the devised problem model are:

- Capability constraints (robot heterogeneity)
- Precedence constraints
- Synchronization constraints
- Time window constraints
- Capacity constraints

4.2 Preliminaries - Vehicle Routing Problem

Our formulation follows the vehicle routing problem model, with specific modifications to fit our needs. *Vehicle Routing Problem* (VRP) is one of the most studied and challenging problems of combinatorial optimization. The interest in VRP is motivated by both its practical relevance and its considerable difficulty. VRP is concerned with obtaining the optimal configuration of routes for a fleet of vehicles to serve a set of customers while minimizing the total cost. Our model relates several variants of VRP, which are named and modeled in the continuation of this section.

4.2.1 The Vehicle Routing Problem

The basic VRP [66] regards a set of nodes $N = \{1, \dots, n\}$ representing n *customers* at different locations and a central *depot* (warehouse), which is usually denoted by 0. Customers are served from one depot by a homogeneous and limited fleet of vehicles. A vehicle serving a customer subset $S \subseteq N$ starts at the depot, travels once to each customer in S, and finally returns to the depot. Each pair of locations (i, j), where $i, j \in N \cup \{0\}$, and $i \neq j$, is associated with a *travel cost* c_{ij} that is symmetric, $c_{ij} = c_{ji}$.

The most studied version of VRP problems is the *Capacitated Vehicle Routing Problem* (CVRP). CVRP is the most widely used VRP variant due to its numerous practical applications in transportation, distribution, and logistics. Essentially, CVRP is a problem where vehicles with *limited payloads* need to pick up or deliver items at different locations. The items have a quantity, such as weight or volume, and the vehicles have a maximum capacity that they can carry. The problem is to pick up or deliver the items at the lowest cost without exceeding the vehicle capacity.

In the CVRP, each customer is assigned a *demand* $q_i, i \in N$ that corresponds to the quantity (e.g., weight or volume) of goods to be delivered from the depot to the customer. There is a set of *vehicles*, $K = \{1, \dots, m\}$, with capacity $Q > 0$, operating at identical cost. In the case of a heterogeneous fleet, the capacity Q is specifically defined for each vehicle (or type of vehicle).

A *route* is a sequence $r^* = (i_0, i_1, \dots, i_s, i_{s+1})$ with $i_0 = i_{s+1} = 0$, and $S = \{i_1, \dots, i_s\} \subseteq N$ is the set of visited customers. The route r^* has cost $c(r^*) = \sum_{p=0}^{s} c_{i_p i_{p+1}}$. A route is considered feasible if the capacity constraint $q(S) \coloneqq \sum_{i \in S} q_i \leq Q$ holds and no customer is visited more than once, $i_j \neq i_k$ for all $1 \leq j < k \leq s$. In this case, the set $S \subseteq N$ is considered a *feasible cluster*.

A solution of a CVRP consists of $m = |K|$ feasible routes[†], one for each vehicle $k \in K$. $|K|$ represents the cardinality of the set K. Therefore, the routes $r_1^*, r_2^*, \dots, r_m^*$ corresponding to the specific clusters $S_1, S_2, \dots, S_m$ represent a feasible solution of the CVRP if all routes are feasible and the clusters form a partition of N.

The given model can be represented by an undirected or directed graph. Let $V = \{0\} \cup N$ be the set of vertices (or nodes). In the symmetric case, i.e., if the cost of moving between i and j does not depend on the direction, the underlying graph $G = (V, E)$ is complete (each pair of graph vertices is connected by an edge) and undirected with edge set

[†]Note that not all vehicles may be required in a solution, and a route can have cardinality 0 (be empty), but it is still a feasible route and therefore $m = |K|$ always.

$E = \{e = (i,j) = (j,i) : i,j \in V, i \neq j\}$ and edge cost c_{ij} for $(i,j) \in E$. Otherwise, if at least one pair of vertices $i,j \in V$ has asymmetric cost $c_{ij} \neq c_{ji}$, then the underlying graph is a complete digraph. We are concerned with the former.

Definition 6 CVRP model formulation. *One of the most common mathematical representations of the VRP model is the* Mixed-Integer Linear Programming *(MILP) formulation [66]. The binary decision variable x_{ijk} is defined to indicate whether the vehicle $k, k \in K$ traverses an edge $(i,j) \in E$ in a given solution. Therefore, the integer linear programming model for the CVRP can be considered as written:*

$$\textit{(CVRP)} \qquad min \sum_{k \in K} \sum_{(i,j) \in E} c_{ij} x_{ijk} \tag{4.1.1}$$

Subject to

$$\sum_{k \in K} \sum_{i \in V, i \neq j} x_{ijk} = 1, \qquad \forall j \in V \setminus \{0\}, \tag{4.1.2}$$

$$\sum_{j \in V \setminus \{0\}} x_{0jk} = 1, \qquad \forall k \in K, \tag{4.1.3}$$

$$\sum_{i \in V, i \neq j} x_{ijk} = \sum_{i \in V} x_{jik}, \qquad \forall j \in V, k \in K, \tag{4.1.4}$$

$$\sum_{i \in V} \sum_{j \in V \setminus \{0\}, j \neq i} q_j x_{ijk} \leq Q, \qquad \forall k \in K, \tag{4.1.5}$$

$$\sum_{k \in K} \sum_{i \in S} \sum_{j \in S, j \neq i} x_{ijk} \leq |S| - 1, \qquad \forall S \subseteq N, \tag{4.1.6}$$

$$x_{ijk} \in \{0,1\}, \qquad \forall k \in K, (i,j) \in E. \tag{4.1.7}$$

The objective function (4.1.1) minimizes the total travel cost. The constraints (4.1.2) are the degree constraints that ensure that exactly one vehicle visits each customer. The constraints (4.1.3) and (4.1.4) guarantee that each vehicle leaves the depot only once, and that the number of vehicles arriving at each customer and returning to the depot is equal to the number of vehicles departing from that node. Capacity constraints are expressed in (4.1.5), and ensure that the sum of the demands of the customers visited on a route is less than or equal to the capacity of the vehicle providing the service. The sub-tour elimination constraints (4.1.6) ensure that the solution does not contain cycles disconnected from the depot. The constraints (4.1.7) specify the domains of the variables. This model is known as a three-index vehicle flow formulation.

4.2.2 The VRP with Time Windows

Vehicle Routing Problem with Time Windows *(VRPTW)* [66, 67] is the extension of capacitated VRP (CVRP), where service must begin at each customer within an associated time interval called a *time window*. Time windows can be hard or soft. In the case of a hard time window, if a vehicle arrives at the site earlier than specified, it must wait for the time window to begin in order to perform service. Later arrival time than specified by the constraint is not admitted in a valid solution. Typically, waiting before the start of a service is a zero-cost action. In contrast, during soft time windows, a vehicle may act outside of the specified time frame, incurring some penalty costs.

Adding to the previously defined model of CVRP, in VRPTW model, time window is associated with nodes $i \in V$ as $[\alpha_i, \beta_i]$, where α_i and β_i denote node i's earliest (latest) service start time. Besides the travel cost (c_{ij}) for each edge $(i, j) \in E$, this model also includes the *travel time* t_{ij}, with symmetry property, $t_{ij} = t_{ji}, (i, j) \in E$. Moreover, service times σ_i are defined for all nodes $i \in V$. They define the duration of processing each node. In a case of the depot, zero service time is assigned, $\sigma_0 = 0$. Feasible solution exists only if $\alpha_0 \leq \min_{i \in V \setminus \{0\}} \{\beta_i - t_{0i}\}$ and $\beta_0 \geq \max_{i \in V \setminus \{0\}} max\{\alpha_0 + t_{0i}, \alpha_i\} + \sigma_i + t_{i0}$ [66].

Definition 7 VRPTW model formulation. *The MILP representation of VRPTW considers two types of variables: flow variables $x_{ijk}, (i, j) \in E, k \in K$, which are equal to* 1 *if the edge (i, j) is used by vehicle k, and* 0 *otherwise; and time variables $\omega_{ik}, i \in V, k \in K$, which indicate the start of service at node i when served by vehicle k. The problem of VRPTW is formulated as [66]:*

$$\text{(VRPTW)} \qquad min \sum_{k \in K} \sum_{(i,j) \in E} c_{ij} x_{ijk} \tag{4.2.1}$$

Subject to

$$\sum_{k\in K}\sum_{i\in V, i\neq j} x_{ijk} = 1, \qquad \forall j \in V \setminus \{0\}, \tag{4.2.2}$$

$$\sum_{j\in V\setminus\{0\}} x_{0jk} = 1, \qquad \forall k \in K, \tag{4.2.3}$$

$$\sum_{i\in V, i\neq j} x_{ijk} = \sum_{i\in V} x_{jik}, \qquad \forall j \in V, k \in K, \tag{4.2.4}$$

$$x_{ijk}(\omega_{ik} + \sigma_i + t_{ij} - \omega_{jk}) \leq 0, \qquad \forall k \in K, \tag{4.2.5}$$

$$\alpha_i(\sum_{j\in V, j\neq i} x_{ijk}) \leq \omega_{ik} \leq \beta_i(\sum_{j\in V, j\neq i} x_{ijk}), \qquad \forall i \in V, k \in K, \tag{4.2.6}$$

$$\sum_{i\in V}\sum_{j\in V\setminus\{0\}, i\neq j} q_j x_{ijk} \leq Q, \qquad \forall (i,j) \in E, k \in K, \tag{4.2.7}$$

$$\sum_{k\in K}\sum_{i\in S}\sum_{j\in S, i\neq j} x_{ijk} \leq |S| - 1, \qquad \forall S \subseteq N, \tag{4.2.8}$$

$$x_{ijk} \in \{0,1\}, \qquad \forall k \in K, (i,j) \in E. \tag{4.2.9}$$

The objective function (4.2.1) minimizes the total travel cost. As in the previously defined model of CVRP, VRPTW contains some of the common constraints – the degree constraints (4.2.2), flow constraints (4.2.3, 4.2.4), capacity constraints (4.2.7), and the subtour elimination constraints (4.2.8). In addition, the constraints (4.2.5) and (4.2.6) guarantee the feasibility of the schedule in terms of time and enforce time windows, respectively. Finally, the domains of the variables are specified by (4.2.9).

4.2.3 The VRPTW with Synchronization and Precedence Constraints

The next refinement of the VRP model is the introduction of synchronization and precedence constraints [68]. These constraints are particularly important for any application involving real-world systems. Often, different vehicles are required to perform a task at the same or different locations, and the operations performed by each vehicle must occur either at the same time or with precedence. Therefore, we define the model of Vehicle Routing Problem with Time Windows and Precedence and Synchronization Constraints *(VRPTW-PS)*.

We define a set of precedence-constrained customers as $\Pi = \{(i,j), i,j \in N\}$. Nodes i and j are time-constrained such that service at customer i must finish before service at customer j begins. In the simple case of precedence constraints defined on a single route, the constraint can be formalized as $\omega_i + \sigma_i \leq \omega_j, (i,j) \in \Pi$, where ω_i and ω_j mark the start of service at nodes i and j, respectively, and σ_i denotes the service duration for node

i. For the case where there are constraints between routes, the expression becomes more complex since the routes of all vehicles in the system must be considered. We formulate such a situation as

$$\sum_{k\in K}\sum_{i'\in V} x_{i'ik}(\omega_{ik}+\sigma_i) \leq \sum_{k\in K}\sum_{j'\in V} x_{j'jk}\omega_{jk}, \qquad \forall (i,j)\in \Pi, \tag{4.3}$$

where $\omega_{ik}, i \in V, k \in K$, denotes the start of service at node i when processed by vehicle k. Nodes i' and j' are possible direct predecessors of nodes i and j, respectively, in the routes of the same vehicle. The sum $\sum_{i'\in V} x_{i'ik}$ is equal to 1 for exactly one action i' that precedes i in the route of vehicle k.

Further, we identify a set of synchronization-constrained nodes as $\Sigma = \{(i,j), i,j \in N\}$. Between each pair of nodes (i,j) there is a coordination constraint that requires them to execute simultaneously. In practice, we distinguish between two different constraints, formalized as follows. The first, $\omega_i = \omega_j, (i,j) \in \Sigma$, matches the start time of processing for nodes i and j. The second similarly matches the end times of processing of nodes and is written as $\omega_i + \sigma_i = \omega_j + \sigma_j, (i,j) \in \Sigma$. The latter condition can be truncated to $\sigma_i = \sigma_j, (i,j) \in \Sigma$ since the first terms in the equations are made redundant by the first synchronization condition. As with precedence constraints, we need to extend the defined equations to account for relations between schedules. Therefore, we define synchronization constraints as

$$\sum_{k\in K}\sum_{i'\in V} x_{i'ik}\omega_{ik} = \sum_{k\in K}\sum_{j'\in V} x_{j'jk}\omega_{jk}, \qquad \forall (i,j)\in \Sigma, \tag{4.4}$$

$$\sigma_i = \sigma_j, \qquad \forall (i,j)\in \Sigma, \tag{4.5}$$

where $\omega_{ik}, i \in V, k \in K$, denotes the start of service at node i when processed by vehicle k. In the defined formulation, $\sum_{i'\in V} x_{i'ik}, i, i' \in V$ yields nonzero values for vehicle $k \in K$ only if node i is served in the corresponding route.

Definition 8 VRPTW-PS model formulation. *The MILP representation of VRPTW-PS considers two types of variables: flow variables $x_{ijk}, (i,j) \in E, k \in K$, which are equal to 1 if the edge (i,j) is used by vehicle k, and 0 otherwise; and time variables $\omega_{ik}, i \in V, k \in K$, which indicate the start of service at node i when served by vehicle k. The problem of VRPTW-PS is defined as:*

$$\textit{(VRPTW-PS)} \qquad \min \sum_{k\in K}\sum_{(i,j)\in E} c_{ij}x_{ijk} \tag{4.6.1}$$

Subject to

$$\sum_{k\in K}\sum_{i\in V, i\neq j} x_{ijk} = 1, \qquad \forall j \in V \setminus \{0\}, \tag{4.6.2}$$

$$\sum_{j\in V\setminus\{0\}} x_{0jk} = 1, \qquad \forall k \in K, \tag{4.6.3}$$

$$\sum_{i\in V, i\neq j} x_{ijk} = \sum_{i\in V} x_{jik}, \qquad \forall j \in V, k \in K, \tag{4.6.4}$$

$$x_{ijk}(\omega_{ik} + \sigma_i + t_{ij} - \omega_{jk}) \leq 0, \qquad \forall k \in K, (i,j) \in E, \tag{4.6.5}$$

$$\alpha_i(\sum_{j\in V, j\neq i} x_{ijk}) \leq \omega ik \leq \beta_i(\sum_{j\in V, j\neq i} x_{ijk}), \qquad \forall i \in V, k \in K, \tag{4.6.6}$$

$$\sum_{k\in K}\sum_{i'\in V} x_{i'ik}(\omega_{ik} + \sigma_i) \leq \sum_{k\in K}\sum_{j'\in V} x_{j'jk}\omega_{jk}, \qquad \forall (i,j) \in \Pi, \tag{4.6.7}$$

$$\sum_{k\in K}\sum_{i'\in V} x_{i'ik}\omega_{ik} = \sum_{k\in K}\sum_{j'\in V} x_{j'jk}\omega_{jk}, \qquad \forall (i,j) \in \Sigma, \tag{4.6.8}$$

$$\sigma_i = \sigma_j, \qquad \forall (i,j) \in \Sigma, \tag{4.6.9}$$

$$\sum_{i\in V}\sum_{j\in V\setminus\{0\}, i\neq j} q_j x_{ijk} \leq Q, \qquad \forall (i,j) \in E, k \in K, \tag{4.6.10}$$

$$\sum_{k\in K}\sum_{i\in S}\sum_{j\in S, i\neq j} x_{ijk} \leq |S| - 1, \qquad \forall S \subseteq N, \tag{4.6.11}$$

$$x_{ijk} \in \{0,1\}, \qquad \forall k \in K, (i,j) \in E. \tag{4.6.12}$$

The objective function (4.6.1) minimizes the total travel cost. This model incorporates many of the properties of the previously defined VRPTW model with capacity constraints – the degree constraints (4.6.2), flow constraints (4.6.3, 4.6.4), capacity constraints (4.6.10), subtour elimination constraints (4.6.11), schedule feasibility (4.6.5), and time window constraints (4.6.6). The limitations defined by (4.6.7) enforce precedence constraints, while synchronization constraints are given by (4.6.8) and (4.6.9). The domains of the variables are specified by (4.6.12).

4.2.4 The VRP with multiple depots

Multi-Depot Vehicle Routing Problem *(MDVRP)* concerns a VRP variant where more than one depot is available to provide service to customers. In principle, each vehicle $k \in K$ can have individual start and end locations. However, in the MDVRP, groups of vehicles are typically assigned to a (smaller) number of depots. Depots may have a limited capacity and may host a limited or unlimited subfleet. Here we consider depots with a limited number of homogeneous vehicles. To formulate this problem, we build on the previously defined CVRP and introduce elements from the MDVRP formulated in [69].

We introduce two different types of vertices of the underlying graph $G = (V, E)$, namely a subset of customer nodes $V_c = N = \{1, \ldots, n\}$ and depot nodes $V_d = \{1, \ldots, l\}$, where l denotes the number of available depots. It holds $V = V_c \cup V_d$ and $V_c \cap V_d = \emptyset$. Each depot $i \in V_d$ is associated with a fleet of vehicles, $K_i \subseteq K$. Therefore, the set of vehicles $K = \{1, \ldots, m\}$ contains a union of the individual depot fleets, $K = K_1 \cup \ldots \cup K_l = \bigcup_{i \in V_d} K_i$. The underlying graph $G = (V, E)$ is complete and undirected with edge set $E = \{e = (i, j) = (j, i) : i, j \in V, i \neq j\}, i$ and j not both in V_d and edge cost c_{ij} for $(i, j) \in E$.

Definition 9 MDVRP model formulation. *Here we present the MILP formulation of the capacitated VRP with multiple depots. The binary decision variable x_{ijk} is defined to indicate whether the vehicle $k, k \in K$ traverses an edge $(i, j) \in E$ in an optimal solution. Therefore, the model is expressed as:*

$$\text{(MDVRP)} \qquad min \sum_{k \in K} \sum_{(i,j) \in E} c_{ij} x_{ijk} \tag{4.7.1}$$

Subject to

$$\sum_{k \in K} \sum_{i \in V, i \neq j} x_{ijk} = 1, \qquad \forall j \in V_c, \tag{4.7.2}$$

$$\sum_{j \in V_c} x_{ijk} = 1, \qquad \forall i \in V_d, k \in K_i \tag{4.7.3}$$

$$\sum_{i \in V, i \neq j} x_{ijk} = \sum_{i \in V} x_{jik}, \qquad \forall j \in V, k \in K, \tag{4.7.4}$$

$$\sum_{i \in V} \sum_{j \in V_c, j \neq i} q_j x_{ijk} \leq Q, \qquad \forall k \in K, \tag{4.7.5}$$

$$\sum_{k \in K} \sum_{i \in S} \sum_{j \in S, j \neq i} x_{ijk} \leq |S| - 1, \qquad \forall S \subseteq N, \tag{4.7.6}$$

$$x_{ijk} \in \{0, 1\}, \qquad \forall k \in K, (i, j) \in E. \tag{4.7.7}$$

The objective function (4.7.1) minimizes the total travel cost. As in the CVRP model defined earlier, the MDVRP includes some of the usual constraints – the degree constraints (4.7.2), flow constraints (4.7.3, 4.7.4), capacity constraints (4.7.5), and the subtour elimination constraints (4.7.6). However, we consider more than one depot in this model, which is especially noticeable in the constraint (4.7.3). Finally, the domains of the variables are specified by (4.7.7).

4.2.5 The Heterogeneous Fleet VRP

The class of Heterogeneous Fleet Vehicle Routing Problem *(HFVRP)* [70] considers groups or types of vehicles that may differ in capacity, cost, speed, and the customers they can reach. The fleet K is partitioned into $|P|$ subsets of homogeneous vehicles $K = K^1 \cup K^2 \cup \ldots \cup K^{|P|}$ (also called vehicle types). All vehicles $k \in K^p$ of type p $(p = 1, \ldots, |P|)$ are characterized by a capacity $Q_k = Q^p$, variable routing costs $c_{ijk} = c^p_{ij}$, and subset $N_k = N^p \subseteq N$ of reachable customers.

It is relatively straightforward to modify the CVRP model to account for vehicle-specific characteristics. We replace the general coefficients with individual ones, e.g., capacity Q with Q_k. Vehicle-dependent routing costs arise by replacing c_{ij} with individual c_{ijk} for all $(i, j) \in E$. To model inaccessible customers $j \in N \setminus N_k$, we set c_{ijk} for all $(i, j) \in E$ to a sufficiently large number M.

Definition 10 HFVRP model formulation. *Here we present the MILP formulation of the capacitated VRP with heterogeneous fleet (HFVRP). The binary decision variable x_{ijk} is defined to indicate whether the vehicle $k, k \in K$ traverses an edge $(i, j) \in E$ in an optimal solution. Then the model is given as:*

$$\textit{(HFVRP)} \qquad \textit{min} \sum_{k \in K} \sum_{(i,j) \in E} c_{ijk} x_{ijk} \tag{4.8.1}$$

Subject to

$$\sum_{k \in K} \sum_{i \in V, i \neq j} x_{ijk} = 1, \qquad \forall j \in V \setminus \{0\}, \tag{4.8.2}$$

$$\sum_{j \in V \setminus \{0\}} x_{0jk} = 1, \qquad \forall k \in K, \tag{4.8.3}$$

$$\sum_{i \in V, i \neq j} x_{ijk} = \sum_{i \in V} x_{jik}, \qquad \forall j \in V, k \in K, \tag{4.8.4}$$

$$\sum_{i \in V} \sum_{j \in V \setminus \{0\}, j \neq i} q_j x_{ijk} \leq Q_k, \qquad \forall k \in K, \tag{4.8.5}$$

$$\sum_{k \in K} \sum_{i \in S} \sum_{j \in S, j \neq i} x_{ijk} \leq |S| - 1, \qquad \forall S \subseteq N, \tag{4.8.6}$$

$$x_{ijk} \in \{0, 1\}, \qquad \forall k \in K, (i, j) \in E. \tag{4.8.7}$$

The objective function (4.8.1) minimizes the total travel cost given the specificity of each vehicle, c_{ijk}. The HFVRP contains the same constraints as the previously defined CVRP – the degree constraints (4.8.2), flow constraints (4.8.3, 4.8.4), capacity constraints (4.8.5), and the subtour elimination constraints (4.8.6). However, in this model, we consider a *specific capacity of each vehicle, Q_k in (4.8.5). Finally, the domains of the variables are specified by (4.8.7).*

4.3 Modeling of Task Planning Problems as VRP

Having defined the underlying mathematical model, we relate it to the task planning problem specified at the beginning of this section. We consider this metaphor at a high level of abstraction, although in some applications involving transportation scenarios the relationship may be more apparent. Here we outline all the features of the proposed model in the context of task planning and finally present the full integrated model formulation. The task planning model metaphor considers a heterogeneous fleet multi-depot VRP variant with precedence and synchronization constraints and time windows, referred in the rest of the section as HFVRP-TWPS.

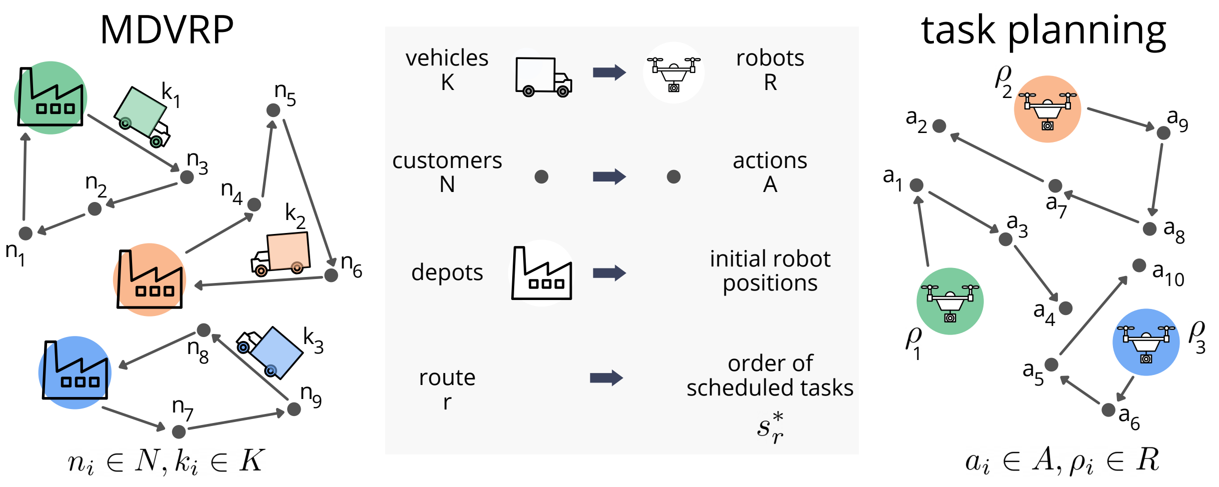


Figure 4.5: An illustration of the relationship of the HFVRP-TWPS model to the task planning paradigm. The diagram shows the direct relation between concepts in VRP and task planning problems, where depots represent initial robot states, vehicles stand for robots, customers refer to actions, and vehicle routes represent final robot schedules.

In the modeling of task planning as a VRP variant, we use the term *customer* for the simple single-robot tasks, *actions*. We designate the set $N = \{1, \ldots, n\} = A$, where A is the set of actions. This definition includes only actionable elements from the robot's task structure, and all tasks must be decomposed down to the action level where vehicle routing is optimized.

In the task planning paradigm, we directly equate the concept of a *vehicle* with *robots*, $K = \{1, \ldots, m\} = R$. Typically, the VRP paradigm requires vehicles to start at the depot, serve assigned customers along the route, and return to the depot. In our modeling, we equate the term *depot* with the *initial location* of the robot. Unlike the vehicles in typical

Table 4.1: Task planning elements as a VRP variant.

Task Planning	HFVRP-TWPS	Unified Model Formulation
actions, A	customers, N	A, set of actions
robots, R	vehicles, K	R, set of robots
schedule, s	route, r^*	$s_r^* = (i_0(r), i_1, \dots, i_{\lvert S_r \rvert}), i \in A, r \in R$
start of the action, a_i^s	start of the service at node	$\omega_{ir}, i \in A, r \in R$
action duration, d_i	service duration	$\sigma_{ir}, i \in A, r \in R$
end of the action, a_i^f	end of the service at node	$\omega_{ir} + \sigma_{ir}, i \in A, r \in R$
initial state of the robot	depot	$i_0(r), r \in R$, initial zero-cost action
action setup cost	travel cost	$c_{ijr}, i, j \in \{0\} \cup A, r \in R$
setup time	travel time	$t_{ijr}, i, j \in \{0\} \cup A, r \in R$
energy requirement of the action	customer demand	$q_{ir}, i \in A, r \in R$
robot energy capacity	vehicle capacity	$Q_r, r \in R$

VRP problems, the robot is not required to return to the starting point. Such a variant of VRP is referred to in the literature as *open VRP* [71].

The proposed model distinguishes between two different cost variants. On the one hand, the cost of transitioning between two tasks, *setup cost*, is directly related to the *travel cost* of the HFVRP-TWPS. This cost can include any movement between locations and the possible setup cost between two tasks and is defined as $c_{ijr}, i, j \in \{0\} \cup A, i \neq j, r \in R$. Closely related to the setup cost is the *setup time*, $t_{ijr}, i, j \in \{0\} \cup A, i \neq j, r \in R$, which defines the duration of the robot's setup between two tasks. The second cost variant is the *energy requirement* of the specific action $i \in A$ is defined as $q_{ir}, r \in R$ and characterized as the customer *demand* in HFVRP-TWPS.

According to the specified modeling, the robot *schedules* are constructed based on the *routes* in the HFVRP-TWPS solution. The parallel is evident, since both constructs represent temporally ordered sequences of jobs. In the open VRP paradigm, a route is a sequence $r^* = (i_0, i_1, \dots, i_s)$ with $i_0 = 0$, where 0 denotes a depot node, and the rest of the elements are customer nodes. In our modeling, we take the previously defined concept of a schedule, and map the order of tasks in a schedule to the arranged sequence of tasks $s_r^* = (i_0, i_1, \dots, i_{\lvert S_r \rvert}), i_j \in S_r, j \in \{1, \dots, \lvert S_r \rvert\}, r \in R$, where i_0 is a zero-cost task associated with the robot's starting location, and S_r is a set of scheduled tasks. Temporal elements of the schedule are the task *start time* and the task *duration*, defined respectively as: $\omega_{ir} \coloneqq a_i^s, i \in S_r, r \in R$ and $\sigma_{ir} \coloneqq d_i = a_i^f - a_i^s, i \in S_r, r \in R$.

A schedule for a robot $r \in R$ is considered feasible if the capacity constraint $q_r(S_r) \coloneqq \sum_{i \in S_r} q_{ir} \leq Q_r$ holds, and no task is scheduled more than once, $i_j \neq i_k$ for all $1 \leq j < k \leq \lvert A \rvert$. When it is necessary to perform repetitive tasks, each occurrence of the task should be treated as a separate entity.

All previously defined relations are summarized in the Table 4.1 and additionally illustrated in Figure 4.5. Following the graph representation of the VRP, we define the task planning problem on a graph. This formulation of the problem is the underlying framework of the proposed model, and we express all subsequent concepts on this basis.

4.3.1 Mathematical model formulation

The mathematical representation of the unified task planning model as MILP is based on the graph structure. Let V be the set of vertices (or nodes) consisting of two distinct sets of nodes, the action nodes $V_a = A = \{1, \ldots, n\}$ and the start position nodes $V_s = \{1, \ldots, l\}$, where n and l represent the number of available tasks and robot start positions, respectively. It holds that $V = V_a \cup V_s$ and $V_a \cap V_s = \emptyset$. The underlying graph $G = (V, E)$ is complete and directed with edge set $E = \{e = (i, j) : i, j \in V, i \neq j, i$ and j not both in $V_s\}$.

We model heterogeneity motivated by a mixed fleet variant of MDVRP, in which the set of customers available to the vehicle corresponds in the task planning formulation to a subset of actions A_r that the robot $r \in R$ can perform. To model unavailable tasks $j \in A \setminus A_r$, we set the setup cost c_{ijr} to a sufficiently large number M for all $(i, j) \in E$ related to the unreachable task j. Further aspects of heterogeneity of multi-robot systems are achieved by replacing the general coefficients of MDVRP with robot-specific ones, e.g., the capacity Q by Q_r, the energy requirement q_i by q_{ir} for all $i \in V_a$, and the cost c_{ij} by c_{ijr} for all $(i, j) \in E$.

Modeling constraints

For the given precedence constraints $prec(a_i, a_j)$, $a_i, a_j \in A$ and assuming that tasks $a_i(a_j)$ correspond to nodes $i(j) \in V_a$, we represent the precedence constraint in VRP notation by adding the pair of tasks to the set of constrained tasks $\Pi = \{(i, j), i, j \in V_a\}$. The constraint on the tasks is then

$$\omega_i + \sigma_i \leq \omega_j, (i, j) \in \Pi, \tag{4.9}$$

where ω_i and ω_j mark the beginning of the tasks i and j respectively, and σ_i denotes the duration of task i. We also distinguish particular ω_{ir} as the start time of task i when it is executed by robot $r \in R$, and the corresponding task duration σ_{ir}. The expressions for these cases are defined in the full model representation.

Synchronization constraints are specified analogously to precedence. For specified $sync(a_i, a_j)$, $a_i, a_j \in A$ and the underlying graph $G = (V, E)$ with corresponding members i and j in the vertex subset V_a, we introduce the pair into the set of synchronization constraints $\Sigma = \{(i,j), i,j \in V_a\}$. The constraints on the tasks are formalized as

$$\omega_i = \omega_j, (i,j) \in \Sigma \tag{4.10}$$

$$\sigma_i = \sigma_j, (i,j) \in \Sigma. \tag{4.11}$$

The constraints ensure that the start times of the tasks are synchronized and that they take the same time to execute, essentially ensuring synchronization. As in the precedence constraint case, in the full model we observe robot-specific time variables ω_{ir}, the start time of task i when executed by robot $r \in R$, and the duration of task i when executed by r, σ_{ir}.

In task planning, we represent a time window for a task $a_i, \in A$ as $tw(a_i, a_i^{s*}, a_i^{f*})$, where a_i^{s*} and a_i^{f*} stand for the earliest start time and latest finish time of the task a_i, respectively. In the VRP paradigm, the concept is similar, usually with a time window that only constrains the start time of the service. In our modeling, we adhere to the task planning model defined earlier and define time windows for the entire duration of a given task. Therefore, for $a_i \in A$ corresponding to node $i \in V_a$, we denote the time window as $[\alpha_i, \beta_i]$, where α_i and β_i represent the earliest service start time and the latest service end time, respectively. The constraining expressions from Equation 4.2.6 are then split into two constraints:

$$\alpha_i \leq \omega_i \leq \beta_i, \tag{4.12.1}$$

$$\omega_i + \sigma_i \leq \beta_i. \tag{4.12.2}$$

Equation 4.12.1 ensures that the start of task execution is within the time window boundaries, while Equation 4.12.2 enforces that the end time of execution is before the end of the specified time frame. In the full model, we consider robot-specific time variables ω_{ir}, the start time of task i when executed by robot $r \in R$, and the duration of task i when executed by r, σ_{ir}.

The optimization objectives

We introduced VRPs as pure routing cost minimization problems. However, in our model there are several aspects that we consider in determining the optimality of the solutions provided. As mentioned earlier, in the task planning section of this chapter, we evaluate tasks and schedules based on three properties – *quality*, *duration*, and *cost*, which correspond to the three objectives in the multi-objective optimization. Task quality is a subjective measure related to the application and is determined a priori by the system designer, while each robot evaluates the duration and cost of a future task with respect to the current state of the system. We specify the task quality as $k_i, i \in V_a$. We have previously defined the robot-specific task duration as $\sigma_{ir}, i \in V_a, r \in R$ and the energy demand as $q_{ir}, i \in V_a, r \in R$, which represents the task cost in this context. Then, the total solution score in these three dimensions can be defined as

$$\kappa = \sum_{r \in R} \sum_{(i,j) \in E} x_{ijr} k_j, \tag{4.13.1}$$

$$\delta = \sum_{r \in R} \sum_{(i,j) \in E} x_{ijr} (t_{ijr} + \sigma_{jr}), \tag{4.13.2}$$

$$\gamma = \sum_{r \in R} \sum_{(i,j) \in E} x_{ijr} (c_{ijr} + q_{jr}), \tag{4.13.3}$$

where the binary decision variable x_{ijr} is defined to indicate if the robot $r, r \in R$ traverses an edge $(i, j) \in E$ in a given solution and t_{ijr} is the setup time between tasks i and j.

In addition, there are several constraints on the system that must hold in the feasible solution, namely capacity, precedence, time window, and synchronization constraints. We consider these constraints as *hard constraints*. However, some temporal constraints may not always hold, but we would like to penalize the solutions that do not adhere. These constraints are considered *soft constraints* and affect the optimality of the particular solution by affecting the value of the objective function. For example, the idle or waiting time of robots in the schedule can be considered a penalty for the solution. We generally accumulate all violated soft constraints into a penalty function π in the evaluation function. This function represents another objective to be minimized for the whole optimization method.

Unified model of task planning as VRP

After defining the translation of the task planning to the VRP model through its essential components and limitations on the system, we present the general model unifying the two paradigms. For convenience, all sets, variables, and constants of the model are summarized in Tables 4.2 and 4.3.

Definition 11 Task planning problem as HFVRP-TWPS formulation. *Here we present the general model of task planning represented in VRP notation, formulated as MILP. The binary decision variable x_{ijr} is defined to indicate if the robot $r, r \in R$ traverses an edge*

Table 4.2: Defined sets in the unified model of task planning as VRP.

Set	Definition
R	set of robots
$V = V_s \cup V_a$	set of vertices (nodes)
V_s	set of robot initial states
$V_a = A$	set of single-robot actions
$E = V \times V$	set of edges
R_i	set of robots starting from location $i, i \in V_s$
Π	set of precedence constraints
Σ	set of synchronization constraints

Table 4.3: Defined variables and constants in the unified model of task planning as VRP.

Variable	Definition	Domain
x_{ijr}	if robot r traverses edge $(i, j) \in E$	$\{0, 1\}$
ω_{ir}	start time of action $i \in V_a$ performed by $r \in R$	$\mathbb{R}$
Constant		
κ_i	quality of action $i \in V_a$	$\mathbb{R}$
σ_{ir}	duration of action $i \in V_a$ when performed by robot $r \in R$	
t_{ijr}	setup time between actions i and j, $\forall(i, j) \in E$ assigned to $r \in R$	
c_{ijr}	setup cost between actions i and j, $\forall(i, j) \in E$ assigned to $r \in R$	
q_{ir}	energy demand of action $i \in V_a$ for robot $r \in R$	
Q_r	energy capacity of robot $r \in R$	
α_i	beginning of time window for action $i \in V_a$	
β_i	end of time window for action $i \in V_a$	

$(i, j) \in E$ in a given solution. Then, the model is given as:

$$\text{maximize} \qquad \kappa = \sum_{r \in R} \sum_{(i,j) \in E} x_{ijr} k_j \tag{4.14.1}$$

$$\text{minimize} \qquad \begin{cases} \delta = \sum_{r \in R} \sum_{(i,j) \in E} x_{ijr}(t_{ijr} + \sigma_{jr}) \\ \gamma = \sum_{r \in R} \sum_{(i,j) \in E} x_{ijr}(c_{ijr} + q_{jr}) \\ \pi \end{cases} \tag{4.14.2}$$

Subject to

$$\sum_{r \in R} \sum_{i \in V, i \neq j} x_{ijr} \leq 1, \qquad \forall j \in V_a, \tag{4.14.3}$$

$$\sum_{j \in V_a} x_{ijr} \leq 1, \qquad \forall i \in V_s, r \in R_i, \tag{4.14.4}$$

$$x_{ijr}(\omega_{ir} + \sigma_{ir} + t_{ijr} - \omega_{jr}) \leq 0, \qquad \forall r \in R, (i, j) \in E, \tag{4.14.5}$$

$$\alpha_i(\sum_{j \in V, j \neq i} x_{ijr}) \leq \omega_{ir} \leq \beta_i(\sum_{j \in V, j \neq i} x_{ijr}), \qquad \forall i \in V_a, r \in R, \tag{4.14.6}$$

$$\omega_{ir} + \sigma_{ir} \leq \beta_i(\sum_{j \in V, j \neq i} x_{ijr}), \qquad \forall i \in V_a, r \in R, \tag{4.14.7}$$

$$\sum_{r \in R} \sum_{i' \in V} x_{i'ir}(\omega_{ir} + \sigma_{ir}) \leq \sum_{r \in R} \sum_{j' \in V} x_{j'jr}\omega_{jr}, \qquad \forall (i, j) \in \Pi, \tag{4.14.8}$$

$$\sum_{r \in R} \sum_{i' \in V} x_{i'ir}\omega_{ir} = \sum_{r \in R} \sum_{j' \in V} x_{j'jr}\omega_{jr}, \qquad \forall (i, j) \in \Sigma, \tag{4.14.9}$$

$$\sum_{r \in R} \sum_{i' \in V} x_{i'ir}\sigma_{ir} = \sum_{r \in R} \sum_{j' \in V} x_{j'jr}\sigma_{jr}, \qquad \forall (i, j) \in \Sigma, \tag{4.14.10}$$

$$\sum_{i \in V} \sum_{j \in V \setminus \{0\}, j \neq i} q_{jr} x_{ijr} \leq Q_r, \qquad \forall r \in R, \tag{4.14.11}$$

$$\sum_{r \in R} \sum_{i \in S} \sum_{j \in S, j \neq i} x_{ijr} \leq |S| - 1, \qquad \forall S \subseteq V_a, \tag{4.14.12}$$

$$x_{ijr} \in \{0, 1\}, \qquad \forall r \in R, (i, j) \in E, \tag{4.14.13}$$

$$\omega_{ir} \in \mathbb{R}, \qquad \forall r \in R, i \in V. \tag{4.14.14}$$

The objectives of the optimization problem are represented by Equations (4.14.1) and (4.14.2). The goal is to find a solution that maximizes the total quality κ while minimizing the overall duration δ, cost γ, and penalties caused by soft constraint violation π. Constraints (4.14.3) require each task to complete at most once. Equations (4.14.4) state that each robot runs at most one schedule. Notice that flow constraints present in previously defined versions of VRP are omitted here. That is because we don't constrain *this model to provide closed-loop solutions as we consider the open VRP formulation. Next, constraints (4.14.5) guarantee schedule feasibility with respect to time considerations, while terms (4.14.6) and (4.14.7) enforce time windows on task start and end times, respectively. The limitations defined by (4.14.8) enforce precedence constraints, while synchronization constraints are dictated by (4.14.9) and (4.14.10). Equations (4.14.11) ensures capacity constraints of robots are met, while (4.14.12) eliminate any possible sub-tours in the solution. Finally, variable domains are provided in (4.14.13) and (4.14.14).*

Example problem

We return to a previously defined use case example of a multi-robot disaster response mission and specify the same problem in the newly defined unified model. In Figure 4.6 is the illustration of the given problem, with the structural parts of the compound mission omitted. In our proposed model, we only regard the action nodes, depicted as rectangles in the graph. First, we identify two backbone sets, set of robots R $=\{$*UGV, humanoid, UAV1, UAV2*$\}$, and the set of actions $A = \{$*clear debris, inspect room, close valve, transport object, monitor operation*$\}$. Assuming that the robots are all deployed from the same location *start*, the set of robot initial states has only one element in it, associated with that node. Therefore, we can define the nodes of the underlying graph as follows:

0 – *start*
1 – *clear debris*
2 – *inspect room*
3 – *close valve*
4 – *transport object*
5 – *monitor the operation*

where the sets of nodes are defined as $V_s = \{0\}$ and $V_a = \{1, 2, 3, 4, 5\}$. Each of the nodes is associated with a real physical location and the values of the setup time t_{ijr}, setup cost c_{ijr} for each robot $r \in R$ and each pair of nodes $i, j \in V_a, i \neq j$ are calculated at the beginning of the mission based on the robot properties and the node locations. At the beginning of the mission, the duration of the action σ_{ir} and the energy demand of the action q_{ir} for each action $i \in V_a$ are also estimated.

The precedence constraints specified in Figure 4.6 are defined as $\Pi = \{(1,\ 2), (3,\ 4), (3,\ 5)\}$. Note that the precedence constraint relating tasks *close valve* and *safe object transport* affects all actions belonging to the compound task of object transportation. Similarly, we define the set of synchronization constraints as $\Sigma = \{(4,\ 5)\}$.

Finally, we present the graph representation of the given problem in the VRP formulation in Figure 4.7. The node for the robot initial states is given as node 0 of the

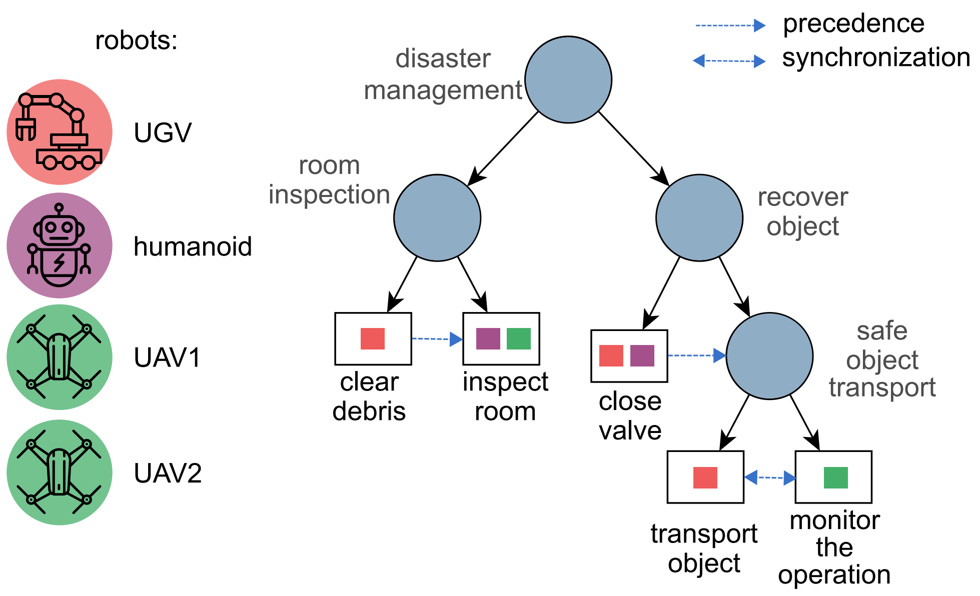


Figure 4.6: An example of a disaster response mission for representation as a VRP-based model. The part of the task structure to be modeled are the action nodes, represented as rectangles.

graph, and its set of robots $R_i = \{UGV,\ humanoid,\ UAV1,\ UAV2\}$. The nodes of the graph are spatially distributed according to their associated task locations. For this simple problem, a solution is proposed, and the routes of the chosen schedule are plotted over the image connecting the robots to their assigned tasks. Then, the specific schedules are generated from the task assignments and sequence by introducing temporal elements into the solution. An example of a schedule is shown in Figure 4.7. Dashed red arrows depict action constraints in the schedule representation.

This very simple problem serves as an illustration of the principles of the proposed modeling. The model can handle very complex missions for multi-robot systems given the multi-objective optimization function. Although solving the model presented in Definition 11 can be solved with an exact solution solver, we choose to implement a fast metaheuristic solution that provides suboptimal solutions in a more efficient way. This solution is presented in the next chapter.

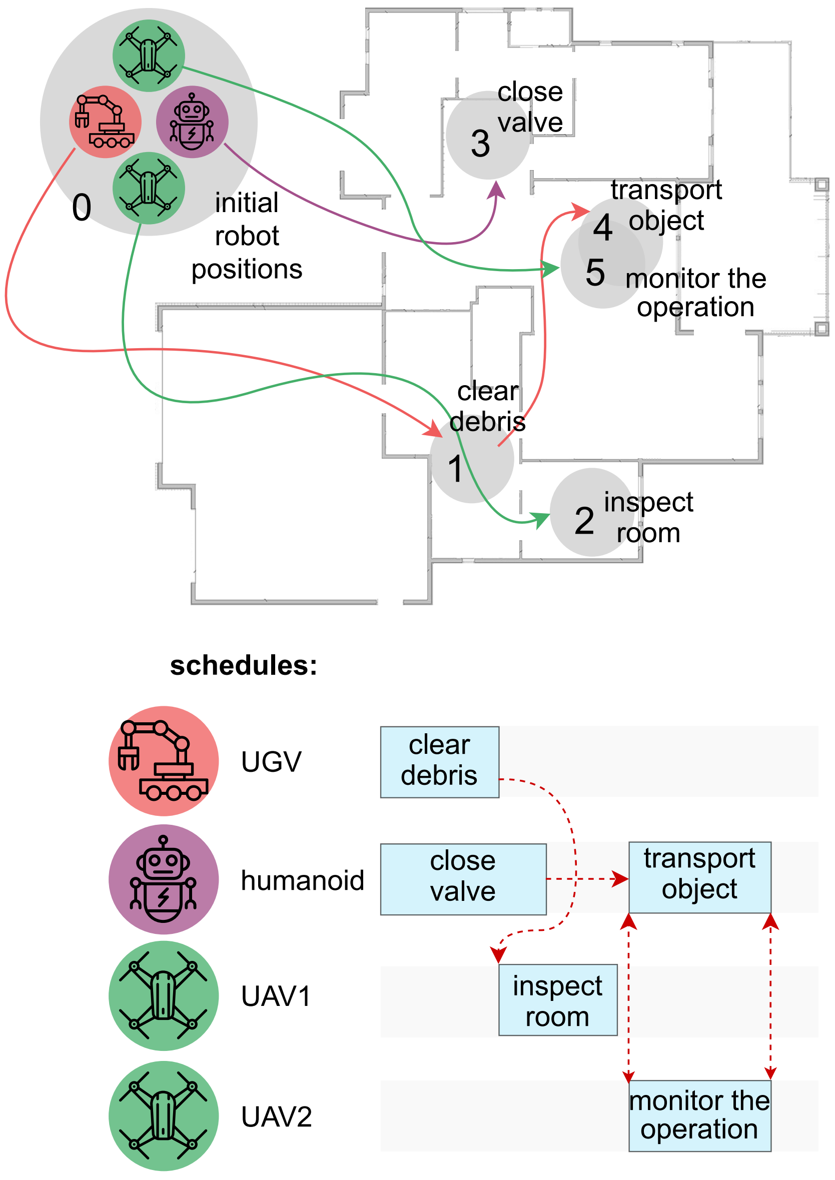


Figure 4.7: Disaster response operation represented as a VRP for spatially distributed tasks. Each action is given as a node $(1, \ldots, 5)$ in the graph. Node 0 represents the initial states of the robots. The schedule for each robot is obtained by introducing temporal elements in the task assignments.

CHAPTER 5

Mission Planning Solution Approach

In this chapter, we describe the proposed solutions to the previously defined mission planning problem for both tasks with cross-schedule dependencies (XD class of problems), as well as for complex tasks (CD class of problems). Both solutions incorporate the use of heuristics and approximations to quickly obtain suboptimal solutions to these very hard combinatorial problems.

5.1 Problems with Cross-Schedule Dependencies

First, in this section, we provide insight into the proposed algorithm for the XD[ST-SR-TA] class of mission planning problems. Based on the problem modeling as a variant of VRP, defined in Section 4.3, we propose a distributed solution inspired by the *Coalition-Based Metaheuristic* (CBM) paradigm.

Similar modeling was proposed in [14], where the problem of task planning refers to the Dial-a-Ride Problem (DARP), a variant of VRP with pickup and delivery. To solve the problem, the authors use a *centralized bounded optimal* branch-and-price algorithm. In our approach, we employ multi-objective optimization with a form of distributed genetic algorithm using mimetism and knowledge sharing. This approach, which uses distributed evolutionary computation methods, can quickly generate near-optimal solutions, and thus, work online while achieving good scalability properties.

5.1.1 MDVRP solution approaches

The problems considered here fit well into the paradigm of the Multi-Depot Vehicle Routing Problem (MDVRP). We regard a MDVRP variant with heterogeneous fleet and capacity constraints (HFVRP-TWPS, as defined in Section 4.3) to incorporate all relevant aspects of task planning. MDVRP is a VRP variation concerned with problems of servicing customers from several depots with a designated fleet of vehicles. It is a classic example of an NP-hard

[5] combinatorial optimization problem. Even for relatively small problem sizes, solving MDVRP to optimality is challenging. Given the rapid combinatorial explosion of factorial dimension, it quickly becomes impossible to obtain optimal solutions for this type of problems.

The authors in [72] were the first to report on optimal solutions for problem sizes up to 50 customers using a branch-and-bound method. The method was soon improved for asymmetric MDVRPs[†] by [73], who first transformed the problem into an equivalent constraint assignment problem and then applied a branch-and-bound technique to problem instances with up to 80 customers and three depots. More recently, [74] developed an exact method for solving the Heterogeneous Fleet Vehicle Routing Problem (HFVRP) that can be used to solve several variants of VRP, including MDVRP. The algorithm is designed around the set partitioning formulation of the problem. It uses three bounding procedures based on the relaxation of the mathematical formulation to reduce the number of variables so that the resulting problem can be solved by an integer linear programming solver. The authors present computational results for MDVRP instances with up to 200 customers and 2-5 depots. The authors in [75] present a new exact algorithm for MDVRP under capacity and route length constraints. The model is defined using a vehicle-flow and a set-partitioning formulation, which are used in different phases of the algorithm. Their method is based on variable fixing, column-and-cut generation, and column enumeration. In their work, optimality has been proven for the first time for some benchmarking instances.

On the other hand, several heuristic methods have been proposed for MDVRP problems. These approaches seek approximate solutions in polynomial time instead of computationally expensive exact solutions. The work in [76] revealed that most researchers tend to solve the MDVRP by heuristics or metaheuristics. In the field of global optimization, *metaheuristics* are stochastic search algorithms specified as generic algorithm frameworks that use rules or heuristics applicable to different problems to accelerate their convergence to near-optimal solutions [77]. In general, metaheuristics emulate processes and behaviors inspired by mechanisms found in nature, such as evolution. One of the most popular metaheuristic algorithms is Genetic Algorithm (GA) [78], and several variants have been proposed to solve MDVRP [79, 80, 81]. One of the main advantages of the GA approaches is their simplicity and fast computation, which allows them to quickly explore a large solution space and effectively generate suboptimal solutions. Other heuristic approaches include Ant Colony Optimization (ACO) algorithm [82, 83], Tabu search [84], and various hybrid methods that combine multiple techniques [85, 86, 87, 88, 89, 90].

The metaheuristics are closely related to multi-agent models, as both can exploit the

[†]The setup cost and time between two nodes depend on the direction of the route.

social metaphor and the self-organization paradigm. Recently, the field of Distributed Artificial Intelligence (DAI) has grown, including multi-agent systems that solve difficult combinatorial problems. The multi-agent concepts can be easily applied to metaheuristics, especially population-based, hybrid, and distributed metaheuristics. The advantages of the distributed approach are the evident increase in computational power due to the simultaneous execution of multiple tasks and the increase in the robustness or efficiency of the search, which is fostered by cooperation and interaction between agents. Our proposed algorithm is inspired by the Coalition-Based Metaheuristic (CBM) approach [91], which uses previously defined principles of DAI to solve the VRP problem.

5.1.2 The Coalition-Based Metaheuristic

In CBM [91], multiple agents organized in a coalition simultaneously explore the solution space, cooperate, and self-adapt to solve the given problem collectively. The novelty introduced in this algorithm was the use of basic DAI principles[†], reinforcement and mimetic learning, which not only allow agents to learn from their experiences and adapt their future behaviors accordingly, but also to share knowledge with other agents in the coalition. In addition to the learned behaviors, the agents also share the best solutions found, so that at the end of each iteration of the algorithm, the best global solution to the problem is obtained.

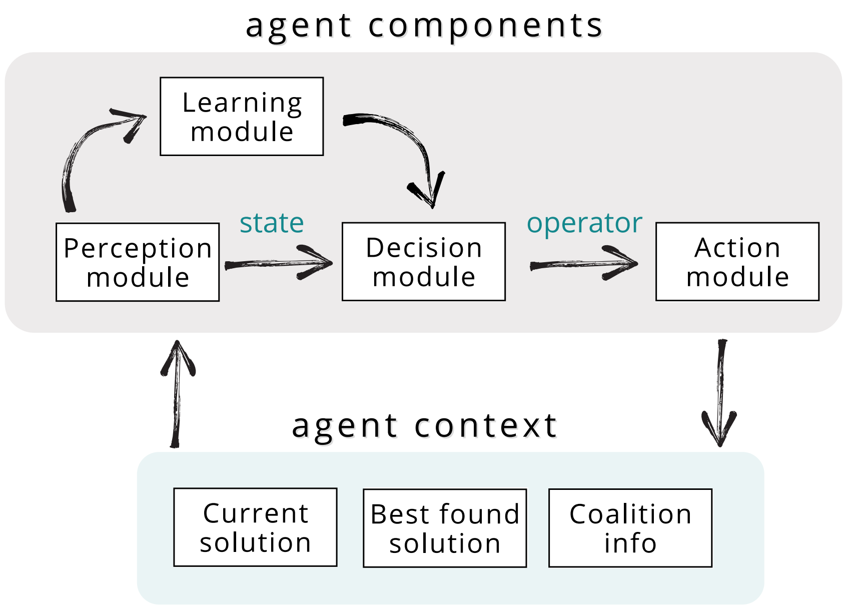


Figure 5.1: CBM agent structure.

The visual representation of the CBM agents is shown in Figure 5.1. During the search process, each agent maintains three solutions, similarly to particle swarm optimization [92]:

[†]autonomous learning processing agents (distributed at large scale and independent) reach conclusions or a semi-equilibrium through interaction and communication

a current solution, the best solution found by the agent, and the best solution found by the entire coalition. An agent uses several operators that are applied to the current solution. The operators can be intensifiers or diversifiers. Intensifier operators concern improvement processes such as local search, and diversifier operators correspond to generation, mutation, or crossover procedures.

The choice of operators to apply is not completely stochastic as in GA. Instead, it is determined by a decision process that uses perceived state and past experience to select the most appropriate operators and coordinate intensification and diversification procedures. The selection of operators is based on heuristic rules. The search behavior of an agent is adapted during the optimization process through an individual reinforcement learning mechanism and mimetic learning. These mechanisms modify the rules of the decision process based on the experience results of previous explorations. Although all agents in the coalition use the same set of operators, the learning mechanisms may ultimately lead to different strategies.

Agents cooperate in two ways. First, an agent can inform the rest of the coalition when it finds a solution that is better than the previous best coalition solution. Second, agents share their internal decision rules to enable mimetic behavior. This fosters search behavior in which desirable solutions are often found.

In [91], the authors proposed the CBM solution for a case of basic VRP. In our case, we consider a HFVRP-PS and thus need to formulate a suitable set of operators. Moreover, we modified the basic CBM algorithm to keep more than one current solution so that a population of solutions is preserved, similar to evolutionary methods. In the rest of the thesis, we refer to the proposed algorithm as *CBM-pop*. Details on the implementation of the algorithm follow in this section.

5.1.3 Distributed metaheuristic for HFVRP-PS

In defining the distributed metaheuristic for task planning problems represented as Heterogeneous Fleet Vehicle Routing Problem with Time Windows and Precedence and Synchronization Constraints (HFVRP-PS), there are some special features that need to be considered. Although our proposed algorithm works very similarly to the CBM algorithm, the problems that these two methods solve are quite different. Here we define the solution representation, the evaluation, the operators used, and the flow of the algorithm for our particular problem.

Solution representation

The first feature to consider in the design of the algorithm is the representation of the solution. Since this metaheuristic is based on a set of operators commonly used in genetic algorithms, we encode solutions in the form of chromosomes. Inspired by an evolutionary process, each chromosome contains genetic material that defines a solution (genotype). In the case of HFVRP-PS, this refers to the assignment of actions to different robots and their order within a sequence of tasks in the schedule. Each chromosome is associated with a phenotype that evaluates the genetic material and, in our case, generates schedules for task sequences based on the temporal properties of the tasks.

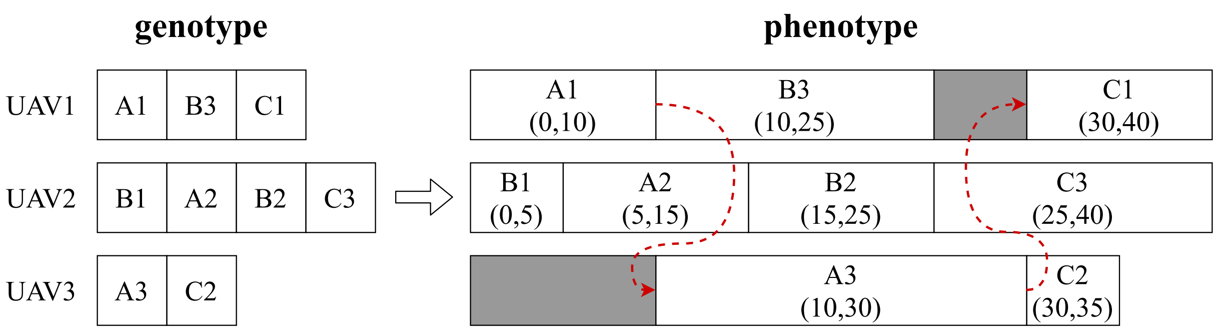


Figure 5.2: Solution representation – chromosome genotype and associated phenotype. The tasks presented here are arbitrarily named generic tasks ($((A, B, C) \times (1, 2, 3))$). The idle times introduced in the schedules are a consequence of precedence constraints $prec(A1, A3)$ and $prec(A3, C1)$, since task $A3$ cannot start before task $A1$ finishes, and task $C1$ cannot start before the end of $A3$.

An example of a chromosome and its genotype and phenotype is shown in Figure 5.2. On the left is shown the genetic material of a solution containing specified task groupings of robots (UAV1, UAV2, UAV3) and ordering. The genotype representation is maintained respecting intra-schedule precedence constraints. On the right is an example of a phenotype generated from the specified genotype. The schedule is formed by introducing time elements into the ordered tasks (task durations, task setup times). If necessary, minimal idle times are inserted to ensure consistency with the defined precedence constraints. The phenotype represents the so-called semi-active schedule, where no left shift is possible in the Gantt graph. For any given sequence of robot operations, there is only one semi-active schedule [93]. One advantage of storing solutions in this way is faster exploration of the solution space, since all operators perform on a simpler genotype representation of the solution. The evaluation procedure renders the phenotype and evaluates the solutions found.

Solution evaluation

The next point to be considered is the evaluation of the solution. Since we are dealing with evaluating solutions based on several different criteria, optimal decisions must be made in the presence of trade-offs between two or more objectives that may be in conflict. There are several possible approaches when considering multiple objectives, the simplest and most widely used is the weighted sum or scalarization method, defined as

$$\text{Minimize}_{\mathbf{x}\in S}\ u(\mathbf{x}) = \sum_{i=1}^{q} w_i f_i(\mathbf{x}), \tag{5.1}$$

where q represents the number of different objectives, S is the set of possible solutions to the problem, and f_i are specific objective functions. The Equation (5.1) represents an optimization problem with a unique objective function $u(\mathbf{x})$. The weights w_i are usually set by the designer of the system, such that $\sum_{i=1}^{q} w_i = 1$ and $w_i \geq 0\ \forall i$.

As simple as it is to apply, this method does have some recognized issues. First, even with some of the methods for determining weights discussed in the literature, satisfactory a priori weight selection does not necessarily guarantee that the final solution obtained will produce solutions that satisfy the original multi-objective problem. The second issue is that it is impossible to obtain points on non-convex parts of the Pareto-optimal set in the criterion space, as will be shown later in the text. This notion ties in with the next approach, namely Pareto-optimality, which is defined in [94] as "*A point x in the feasible design space S is called Pareto-optimal if there is no other point in the set S that reduces at least one objective function without increasing any other*".

Definition 12 *Pareto optimality [94]. A point $\mathbf{x}^* \in S$ is Pareto optimal iff $\nexists$ another point $\mathbf{x} \in S$ such that $f_i(\mathbf{x}) \leq f_i(\mathbf{x}^*)\ \forall i$ and $f_i(\mathbf{x}) < f_i(\mathbf{x}^*)$ for at least one i.*

In Figure 5.3, we illustrate the comparison between the weighted sum approach and the Pareto front. On the left-hand side, we show the case of a convex Pareto front, in which the weighted sum gives satisfactory results. However, on the right-hand side, the figure shows the inability of the scalarization method to approximate the multi-objective function in the case of a non-convex Pareto front. In the graphs, Z represents the feasible criteria space, i.e., the set of objective function values corresponding to feasible points in the design space: $Z = \{f(\mathbf{x})|\mathbf{x} \in S\}$.

Based on these findings, we chose a Pareto ranking procedure [95] for our solution, which assigns ranks to all solutions based on the non-dominance property (i.e., a solution with a lower rank is clearly superior to solutions with a higher rank concerning all objectives). Therefore, solutions are stratified into multiple ranks based on their ability to meet the optimization objectives.

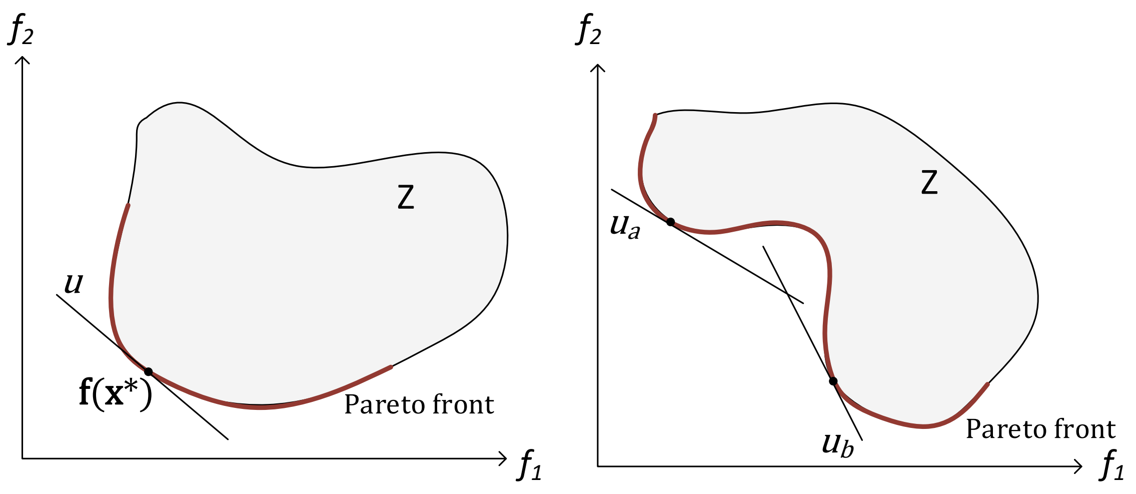


Figure 5.3: Weighted-sum optimization compared to convex (left) and concave (right) Pareto front.

To evaluate solutions in a population P, we apply the double-rank strategy, which takes into account both the density information and the distribution of the solution in the rank. In the first step, an individual $i \in P$ is assigned a dummy rank value $R'(i)$ representing the number of solutions that dominate it in the current population P:

$$R'(i) = |j, j \in P, i \prec j|, \forall i \in P, \tag{5.2}$$

where the symbol $\prec$ corresponds to the Pareto dominance relation, i.e., $i \prec j$ if the solution j performs better than i given all optimization criteria. The final rank of solution $R(i)$ is then defined as the sum of its own dummy rank value and that of its dominators:

$$R(i) = R'(i) + \sum_{j \in P, i \prec j} R'(j), \forall i \in P. \tag{5.3}$$

The second part of the fitness function is the density function, which determines how similar the solution is to the other individuals in the population. Here we use a fairly simple solution where the density of an individual is inversely proportional to the distance to the nearest solution in the population and is calculated as follows:

$$dens(i) = \frac{1}{min(d(i,j), \forall j \in P) + 2}, \forall i \in P, \tag{5.4}$$

where $d(i,j)$ represents the Euclidean distance between two individuals in the criteria space. Finally, the fitness of the solution is obtained as:

$$fitness(i) = \frac{1}{R(i) + dens(i) + 1}, \forall i \in P. \tag{5.5}$$

In this setup, the rank of the solution has a much greater weight in the fitness function, since it is a natural number denoting a subset of solutions from the population, and the rank is a positive number smaller than 1/2. The role of the density function is to discriminate between solutions of the same rank and to favour the more diverse solutions, as they are deemed more likely to explore new regions of the solution space.

Operator definition

Next, we discuss the operators that form the core of the algorithm. We distinguish between generation, diversification (crossover and mutation operators), and intensification operators (local search algorithms). In our application, the generation operator is not used as a diversifier because it is applied during initial population creation. In the proposed solution, we use a single generation operator, a greedy insertion method that randomly takes an unassigned task and inserts it into existing routes at minimal cost, taking into account capacity constraints.

Other operators are listed and described in Table 5.1. Diversification operators are first introduced in [96], and we implemented them for our specific problem. In the crossover procedure, we distinguish two cases of Best-Cost Route Crossover (BCRC), depending on the choice of parent chromosomes. One of the parents is always the current solution of the agent and the second parent is either the best solution found within the whole coalition or selected from the population. Similarly, we adapted the local search algorithms developed in [91] for a VRP problem class. The intensifiers are applied to improve a found solution.

Distributed algorithm definition

The behavior of a single CBM-pop agent is described in Algorithm 2. Before starting the algorithm, the agents exchange their specific problem parameters – task durations σ_{ir}, setup times t_{ijr}, setup costs c_{ijr}, energy demands q_{ir}, and energy capacities Q_r. Precedence and synchronization constraints and time windows are known to all robots since the beginning. During the runtime of the algorithm, the best solutions found and the weight matrices are exchanged among the agents, as noted in Algorithm 2. The procedure itself consists of Diversification-Intensification cycles (D-I cycles), where a diversification operator is first applied to the solution to guide the search out of the local optimum. After this perturbation, a series of local search procedures are applied to the solution to arrive at a new (local) optimum. The process is repeated until a termination criterion is reached.

A major innovation compared to the original CBM is that an agent stores a population of solutions, rather than just one solution. The idea is to further diversify the search and allow broader exploration. The role of the population is twofold. First, after each *n_cycles* cycles without improvement over the best solution found, a new starting solution is

Table 5.1: Genetic operators used in our proposed solution.

Diversifiers [96]	
	Crossover
Best-Cost Route Crossover (BCRC)	For two parent chromosomes, select a route to be removed for each. The removed nodes are inserted into the other parent solution at the best insertion cost.
	Mutation
intra depot reversal	Two cutpoints in the chromosome associated with the robot initial state (depot) are selected and the genetic material between these two cutpoints is reversed.
intra depot swapping	This simple mutation operator selects two random routes from the same initial state (depot) and exchanges a randomly selected action from one route to another.
inter depot swapping	Mutation of swapping nodes in the routes of different initial states (depots). Candidates for this mutation are nodes that are in similar proximity to more than one initial state.
single action rerouting	Re-routing involves randomly selecting one action, and removing it from the existing route. The action is then inserted at the best feasible insertion point within the entire chromosome.
Intensifiers [91]	
two swap	Swapping of borderline actions from two initial states (depots) to improve solution fitness.
one move	Removal of a node from the solution and insertion at the point that maximizes solution fitness.

randomly selected from the population. Second, solutions from the population participate as a second parent in the crossover operator, thereby introducing novelty from the genetic pool.

As the algorithm shows, the decisions of each agent depend on the current state of the algorithm. The states can be defined arbitrarily for each application scenario. In our application, we consider the following set of states $\Gamma = \{\gamma_1, \ldots, \gamma_5\}$, where each state stands for:

- γ_1 – crossover operator was applied
- γ_2 – mutation operator was applied
- γ_3 – intensification operator *one move* was applied
- γ_4 – intensification operator *two swap* was applied
- γ_5 – all intensification operators were applied without changing the solution.

Given the set of states Γ and the set of operators O, the weight matrix W is defined on the domain $\Gamma \times O$. The weight $w_{i,j} \in W$ corresponds to the probability of choosing the operator $o_j \in O$ in the state $\gamma_i \in \Gamma$. The effective choice of an operator follows the principle of the roulette wheel. Thus, the probability $P(o_j|\gamma_i)$ of applying the operator o_j

Algorithm 2: Population-based CBM algorithm (CBM-pop).

```
input  : $pop\_size$ – number of solutions in population
input  : $\boldsymbol{\eta}$ – reinforcement learning factors
input  : $\rho$ – mimetism rate
input  : $n\_cycles$ – number of cycles before changing exploration origin
input  : $\epsilon$ – minimal solution improvement
output : $c_{best\ coalition}$ – best found solution
/* initialization                                                  */
$P \leftarrow$ generate_population($pop\_size$)
evaluate_population($P$)
$c_{current} \leftarrow$ select_solution($P$)
$W \leftarrow$ init_weight_matrix()
$H \leftarrow$ init_experience_memory()
while stopping criterion is not reached do
    /* calculate current state                                     */
    $\gamma \leftarrow$ perceive_state($H$)
    if no change in best solution $> \epsilon$ for $n\_cycles$ then
        evaluate_population($P$)
        $c_{current} \leftarrow$ select_solution($P$)
    end
    $o \leftarrow$ choose_operator($W$, $\gamma$)
    $c_{new} \leftarrow$ apply_op($o$, $c_{current}$, $P$, [$c_{best\ coalition}$])
    /* update experience history                                   */
    $gain \leftarrow fitness(c_{current}) - fitness(c_{new})$
    update_experience($H$, $\gamma$, $o$, $gain$)
    /* update solutions                                            */
    if $c_{best\ coalition}$ improved then
        broadcast_solution($c_{new}$)
    end
    /* learning mechanisms                                         */
    if end of D-I cycle then
        if $c_{best\ coalition}$ improved in the cycle then
            $W \leftarrow$ individual_learning($W$, $H$, $\boldsymbol{\eta}$)
        else if $c_{best\ agent}$ improved in the cycle then
            $W \leftarrow$ individual_learning($W$, $H$, $\boldsymbol{\eta}$)
            broadcast_weight_matrix($W$)
        end
    end
    if weight matrix received from a neighbor then
        $W \leftarrow$ mimetism_learning($W$, $W_{received}$, $\rho$)
    end
end
```

in the state γ_i is calculated by the following formula.

$$P(o_j|\gamma_i) = \frac{w_{i,j}}{\sum_{k=1}^{m} w_{i,k}}, \tag{5.6}$$

where m denotes the number of operators.

After each operation, the agent updates its experience history H by adding to it an element $(\gamma, o, gain)$ that refers to the outcome of that operation. At the end of each diversification-intensification episode, learning is applied based on the goodness of the solution obtained. If the new solution is better than the best solution the agent has found so far, an *individual learning* procedure updates the agent's experience. Given the episode experience history H, the agent updates its weight matrix W according to the rule

$$w_{i,j} = w_{i,j} + \eta \tag{5.7}$$

for γ_i, o_j related to the elements of experience. In the algorithm, we distinguish two cases, (i) when the agent improves its best found solution, and (ii) when the agent improves the best coalitional solution. For this, we define two learning factors η_1 and η_2, giving more weight to the latter.

Besides individual learning, agents also participate in *mimetism learning*, which occurs whenever an agent discovers the new best coalition solution. At that moment, the agent shares its weight matrix W with the rest of the coalition. Whenever agent a receives a weight matrix from another agent b, mimetism learning is applied as

$$W_a = (1 - \rho)W_a + \rho W_b, \tag{5.8}$$

with the mimetism rate ρ and W_a and W_b the weight matrices of agents a and b, respectively.

The combination of individual and mimetic learning introduces adaptability into the method by allowing agents to explore the solution space more effectively. The operator selection decision is guided by prior knowledge and thus a faster descent into optimal domains is more likely. The reinforcement procedure allows local behavior to be improved. However, mimetic behavior allows to exploit the search strategies developed by the other agents.

5.2 Problems with Complex Dependencies

Building upon the CBM-pop presented in Section 5.1, we now extend the method to account for problems with complex dependencies (CD class). Note that these problems surge when tasks can be multiply decomposable, so an extra step of task decomposition selection is needed to find its solution. The proposed approach involves a multi-stage optimization procedure. First, a heuristic task decomposition selection generates possible task decompositions, and the several best ones are subjected to a metaheuristic allocation and scheduling procedure that outputs the final schedule. The allocation and scheduling procedure is the one defined in Section 5.1.

5.2.1 Heuristic task decomposition selection procedure

The problem of task decomposition selection involves finding a subset of actions and tasks to be performed that are most promising to provide near-optimal schedules. In this step of the procedure, a heuristic tree search algorithm is used to quickly generate alternative subsets of tasks that satisfy the mission objective.

Based on the hierarchical task structure defined earlier (Section 3.1), we define a *task alternative* $alt(t)$, $alt(t) \subseteq A$, $t \in T$ as an unordered set of all actions whose execution leads to the completion of task t. The sets A and T represent sets of simple tasks (cations) and composite tasks (tasks), respectively. The concept of task alternatives comes from the GPGP framework as defined in Section 3.2. The complexity of the graph search procedure depends on the structure of the mission tree and the relationships between the nodes. Adding more constraints to the mission simplifies the procedure by removing several options for executing tasks. However, the combinatorial explosion can lead to a factorial size complexity of the task alternative generation procedure for missions without any node relationships.

The process of generating task alternatives starts at the action nodes of the mission tree and builds up recursively, eventually ending at the root of the tree, as outlined in Algorithm 3. To tame a potential combinatorial explosion, the procedure uses a method of focusing the solution search by pruning the worst partial results at each step of the process to make the problem tractable. During this procedure, the robots use estimated values for quality ($k_a(i)$), duration ($d_a(i)$), and cost ($c_a(i)$) of actions, which are determined as the average of these values for all robots that can perform the action, as follows:

$$(\overline{k}_a, \overline{d}_a, \overline{c}_a) = \sum_{i \in \rho_a} \frac{(k_a(i), d_a(i), c_a(i))}{|\rho_a|}, a \in A. \tag{5.9}$$

ρ_a defines the set of robots that can perform action a, $\rho_a = \{i, i \in R, a \in A_i\}$, $\forall a \in A$. $|\rho_a|$ stands for the cardinality of the set ρ_a. R is the set of all robots.

Finally, the score for each task alternative $alt(t)$ is computed based on the expected values of actions $(\overline{k}_a, \overline{d}_a, \overline{c}_a)$, and given the quality accumulation function Q and the defined tree structure. If we specify the alternative outcome values as $(k_{alt}, d_{alt}, c_{alt})$, our simplified objective function (score) for an alternative is defined as

$$sc(k_{alt}, d_{alt}, c_{alt}) = \alpha k_{alt} - \beta d_{alt} - \gamma c_{alt}, \ \alpha, \beta, \gamma \in \mathbb{R}, \tag{5.10}$$

where $\alpha + \beta + \gamma = 1$, and they represent the importance weighting of each specific criterion.

Algorithm 3: Heuristic alternative generation procedure.

```
parameter : $\mu$ – max alternative number for a single task
input     : $tree$ – hierarchical task tree specifying the mission
input     : $criteria$ – evaluation criteria
output    : $task\_alt$ – alternative decompositions of task $task$
Function generate_alternatives($task$):
    /* recursion stopping criteria                                  */
    if $task \in A$ then
        return $[task]$;
    end
    /* recursively generate task alternatives                       */
    $alt \leftarrow []$;
    for $subtask \in tree.subtasks(task)$ do
        $alt \leftarrow alt \cup$ generate_alternatives($subtask$);
    end
    /* combine subtask alternatives based on function $Q$           */
    switch $tree.Q(task)$ do
        case $AND$ do
            $task\_alt \leftarrow$ cartesian_product($alt$)];
        end
        case $XOR$ do
            $task\_alt \leftarrow [[task \cup alt^*]$ for $alt^* \in alt]$;
        end
    end
    /* pruning procedure                                            */
    if $|task\_alt| > \mu$ then
        $sc \leftarrow$ evaluate_alternative($task\_alt$, $criteria$);
        $top \leftarrow$ index_of_max_n_elements($sc, \mu$);
        $task\_alt \leftarrow [task\_alt[i], i \in top]$;
    end
    return $[task\_alt]$
```

Based on these factors, the planning strategy adjusts and selects the appropriate tasks for execution. Using this evaluation function, we are able to define the importance of each problem parameter in the selection of task decomposition. As a result of this process, the robots are given a set of alternative ways to achieve the mission goal (root task). These alternatives are a direct input for the next phase of the optimization, allocation and scheduling process.

The whole process defined in this chapter is shown in Figure 5.4 in the context of the defined coordination framework (Section 3.2.3). After defining the framework, problem modeling and solution approach, the discussion on application scenarios and performance results of the proposed approach follows in the next chapter.

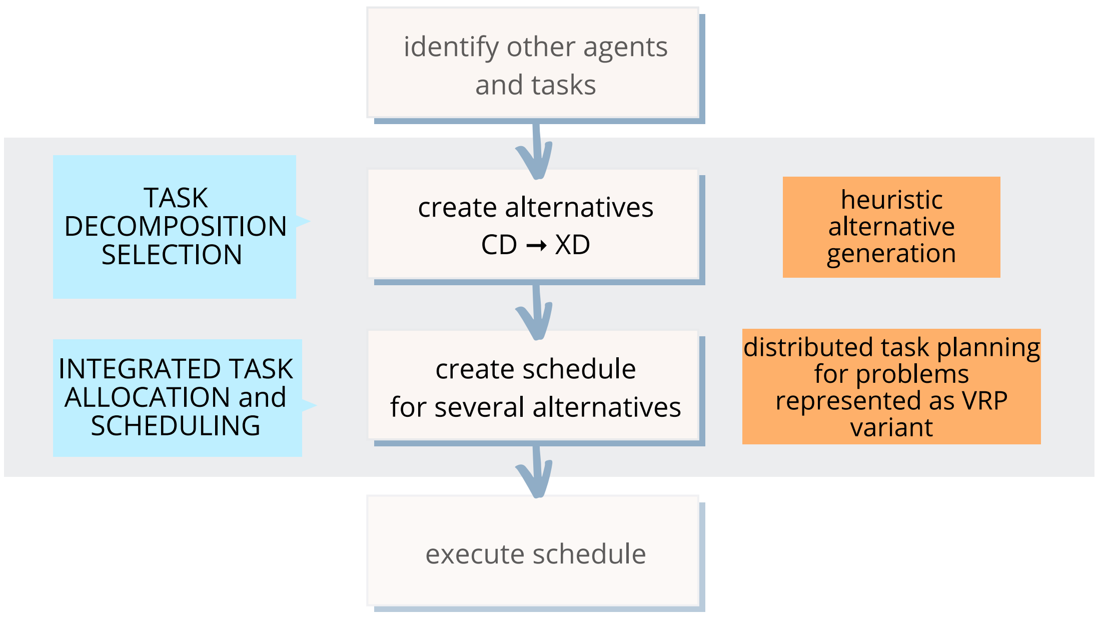

identify other agents and tasks
TASK DECOMPOSITION SELECTION
create alternatives CD → XD
heuristic alternative generation
INTEGRATED TASK ALLOCATION and SCHEDULING
create schedule for several alternatives
distributed task planning for problems represented as VRP variant
execute schedule

Figure 5.4: Mission planning solution in the context of the coordination framework process.

CHAPTER 6

Results and Discussion

In this chapter, we present the results of applying the proposed algorithms and coordination framework to a wide range of heterogeneous multi-robot systems. The results are presented chronologically with respect to the date of the research publication and we can observe the evolution of the methodology throughout.

6.1 Aerial-ground Robotic System for Autonomous Delivery Tasks

In developing the multi-robot mission coordination framework, we start from the custom implementation of the GPGP and its associated modules defined in Section 3.2.1. In [97], we propose a high-level task planning framework based on TÆMS decomposition to extend the capabilities of ground and aerial robots by making them work together. As such, the framework can be used to commission other types of robots as long as they have well-defined capabilities structured into actions that can be integrated into the framework. There are numerous use cases for such a framework, including but not limited to: search and rescue operations, recovery missions, robotic warehouses, etc.

In the previous work which is not described in this thesis [98], we perform a simulation-based verification of the proposed framework in a simple task involving a single *Unmanned Aerial Vehicle* (UAV) and a single *Unmanned Ground Vehicle* (UGV). In this work, we go a step further and experimentally verify an extended version of the framework in a mock-up scenario with two Pioneer UGVs and one UAV equipped with dexterous manipulator arms. The goal of this robotic team is to empty a stack of parcels while negotiating obstacles, using optimization criteria such as speed and energy conservation. The proposed improved task planning algorithm includes applications where a single UAV utilizes both UGVs at different times and locations on the map.

6.1.1 Mission specification

The envisioned application scenario involves an autonomous UAV with a pair of manipulator arms - Unmanned Aerial System (UAS) and UGVs cooperatively working together to transport a parcel between two locations in the environment. Inputs to the described task allocation procedure, which is executed separately on each robot, are the 3D occupancy map of the environment, the source and destination locations of the parcel, and the local view of the TÆMS tree describing the parcel transportation scenario (mission). The global TÆMS tree for the scenario contains 33 action nodes combined with 18 task nodes for a total of 51 nodes. We describe only the main task groups here, as the TÆMS tree is too large to be conveniently included in the text. We use the following term – *task id (robot types that have a task in their TÆMS structure)*. The problems modeled in this example fall under the class XD[ST-SR-TA] in the mission planning problem classification.

The root task *Transport Parcel Mission (UAS, UGV)* consists of two subtasks, *Pick up parcel (UAS, UGV)* and *Transport to destination (UAS, UGV)*, which are associated with the AND operator and the *prec* relation. The second task is further decomposed into three subtasks associated with the XOR operator: *UAS carry alone (UAS)*, *Pass to UGV1 (UAS,UGV)*, and *Pass to UGV2 (UAS,UGV)*. Here, *UGV1* denotes the vehicle closest to the parcel source position, and *UGV2* denotes the vehicle closest to the destination position. The second task is further decomposed to consider the possibility of handing over the parcel to another vehicle nearby with UAS. This is the case when *UGV1* encounters an obstacle while traveling to the destination. In this case, it might be possible to hand over the parcel to another vehicle that is closer to the destination.

The TÆMS tree is suitable for systems with any number of UAS and UGV robots. 33 action nodes correspond to a total of 8 (2) basic UAS (UGV) behaviors, such as "Take off", "Land", "Grab", "Go to position". All behaviors are a kind of motion primitive and their cost and duration are estimated online based on the positions of the parcel and other robots in the environment. The cost is estimated as the product of the path length (we use sampling-based path planners) and the energy consumed per unit of path, which depends on the type of action and the robot involved. The quality of all actions is the same and is set to 1.

6.1.2 Testbed description

To evaluate the developed framework for autonomous task execution, we use three robots: two mobile robots (Pioneer P3DX) and a single 3D Robotics quadrotor equipped with a dual-arm manipulator. The main advantage of using a Pioneer robot is its autonomy time (about 90 minutes) and the ability to carry a parcel and the quadrotor. However, as a mobile robot, it cannot negotiate many ground obstacles. On the other hand, the flight time of the quadrotor is only up to 20 minutes, but it can fly over ground obstacles and carry a parcel using its manipulators. Using the above complementary features of both types of robots, we form a heterogeneous robotic system that can perform autonomous delivery tasks with longer autonomy time and higher energy efficiency than a single robot unit performing the same mission.

The Pioneer P3DX robot is equipped only with its basic sensor technology: encoders and ultrasound sensors (sonars). Since we are using an a priori known static map, sonars are not used. At the software level, the Pioneer computer runs Ubuntu 14.04 with ROS Indigo. Onboard Pioneer, several ROS packages run concurrently to support mission execution - p2os_purdue driver (communication with low-level controller), Move Base package (navigation and position control), action server (interface between navigation stack and high-level planner), and an instance of the developed high-level mission planner (GPGP) described in Section 3.2.1.

The quadrotor carries two control boards - an ArduPilotMega board (APM) for attitude control and an Odroid U3 board with installed Ubuntu and ROS for mission planning, navigation and position control. Another instance of the high-level mission planner runs on an Odroid, along with an action server, a navigation and control stack – *Open Motion Planning Library* (OMPL) extended with a custom trajectory planner, PID position controller, and a roscopter driver for communication with the APM board.

Another important testbed component is the OptiTrack Motion Capture System. Our robotic units use data with their 6-DOF pose information streamed from the Optitrack server. However, during mission planning and execution, each robot uses only its own pose information and neglects the pose data of the other robots in a distributed manner. At the software level, the backbone of our system is ROS, which we use as middleware to exchange data between robots. Since all robots are connected to the same WiFi network with limited bandwidth, we use ROS Rocon Multimaster to limit the data exchange over the network, which in turn allows a higher frequency of Optitrack data (at least $20Hz$).

We run the experiments in the same mock-up arena, first in simulation and then in the real environment, with only minor changes in the software packages. The results are presented in the following sections.

6.1.3 Simulation results

The system described in this section was first thoroughly tested in the Gazebo simulation environment using different criteria specification setups, all of which yielded different task completion scenarios. The simulation testbed used for these tests is shown in Figure 6.1. There are many ways the cooperative robotic system can achieve task completion at the root level, some of which require significant coordination between different robots. For all test scenarios, we used the same values for the flight (driving) cost per unit length, namely $E_{UAV} = 10$ ($E_{UGV} = 1$). In addition, we introduced a different cost rate for the action of the copter carrying the parcel, denoted E'_{UAV}, which is 100. To generate different mission solutions, we varied the relative importance of the quality, duration, and cost components for each test example.

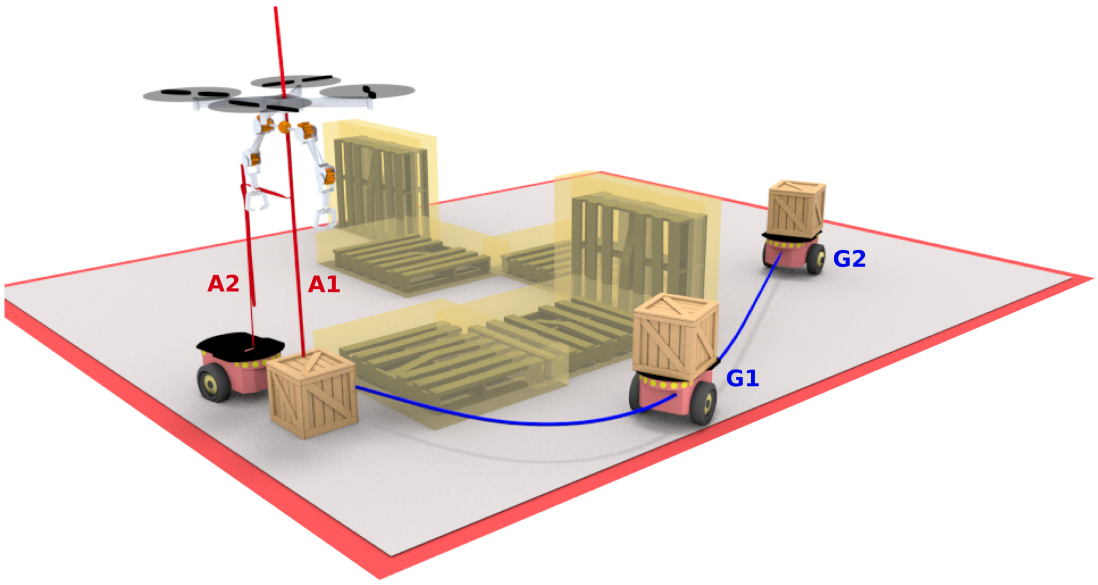


Figure 6.1: Scenario 2: Designed solution where a UAV and a single UGV are used to deliver a parcel. The UAV first grabs the parcel (A1), then flies just above the UGV before dropping the parcel on top of the UGV (A2). The UGV moves (G1) to finally deliver the parcel (G2).

In the first scenario, we give the greatest weight to the duration component of the schedule score $(\alpha, \beta, \gamma) = (0, 100, 0)[\%]$. In other words, we aim for a faster solution, regardless of the resulting cost. The solution to the given coordination problem is also the most obvious one. It consists of the copter picking up the parcel and bringing it to the target position. Due to simplicity and lack of cooperation, simulation results are omitted for this outcome.

In the second scenario, we define a criterion that favors schedules with the lowest cost, where the relative importance of the quality, duration, and cost components is given as $(\alpha, \beta, \gamma) = (0, 0, 100)[\%]$. The second result, shown in Figure 6.1, is that the copter picks up a parcel and loads it onto a Pioneer robot closest to the initial position of the parcel. After the copter releases the load and takes off, Pioneer completes the mission by driving to the target position.

In the third scenario, we intentionally block the passage for the ground vehicle by placing an obstacle in the environment so that the UAS must cooperate with UGVs to achieve the most cost-effective solution. The ratios of quality, duration, and cost are the same as in the previous scenario. The third solution, shown in Figure 6.2, starts similarly

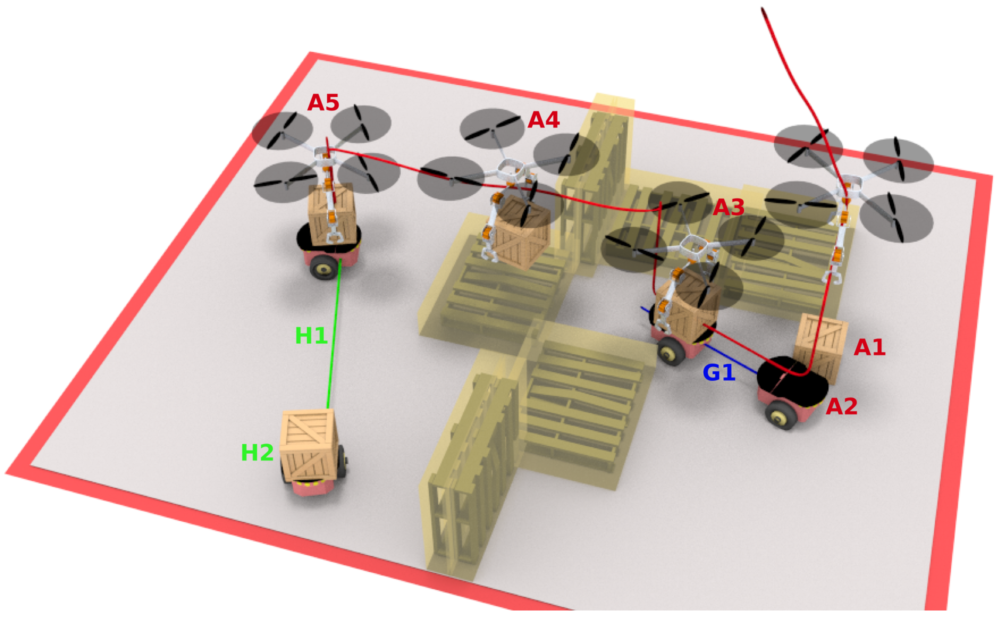


Figure 6.2: Scenario 3: Designed solution where a UAV and two UGVs are used to deliver a parcel. In this scenario, UAV first grasps the parcel (A1), then flies just above UGV1, and finally lands on UGV1 together with the parcel (A2). Next, UGV1 carries UAV and parcel (G1) in front of the next obstacle on the way to the transfer point, where the UAV takes off with parcel (A3). Then, UAV flies over the obstacle (A4) towards UGV2 and deposits the parcel onboard UGV2 (A5). Finally, UGV2 travels with the parcel (H1) and delivers it (H2).

with the copter picking up a parcel and carrying it to the Pioneer robot closest to the initial position of the parcel. After hovering over the Pioneer robot, the copter with the parcel lands on the mobile robot. It then travels at a standstill together with the Pioneer robot until they arrive at the front of the next obstacle. At the same time, the second Pioneer robot starts moving towards the obstacle to meet the first pair of robots for the parcel transfer. When all vehicles meet at the obstacle, the copter transports the parcel from the first to the second Pioneer, which completes the mission by driving the parcel to its target position.

6.1.4 Experimental results

Here we show the results of the selected second scenario, performed in the real-world experiment. Just as in the simulation, the plan is successfully laid out, and its execution is shown in Figure 6.3. We have successfully performed several experiments without mission failures. The motion of the quadrotor, which is crucial for the successful execution of the mission, is shown in Figure 6.4. The video of all performed experiments is available at [99].

In this work, we proposed and tested a decentralized task planning and coordination framework for systems composed of multiple robots with different capabilities. The approach assumes that the map of the area and hierarchical decomposition of the system

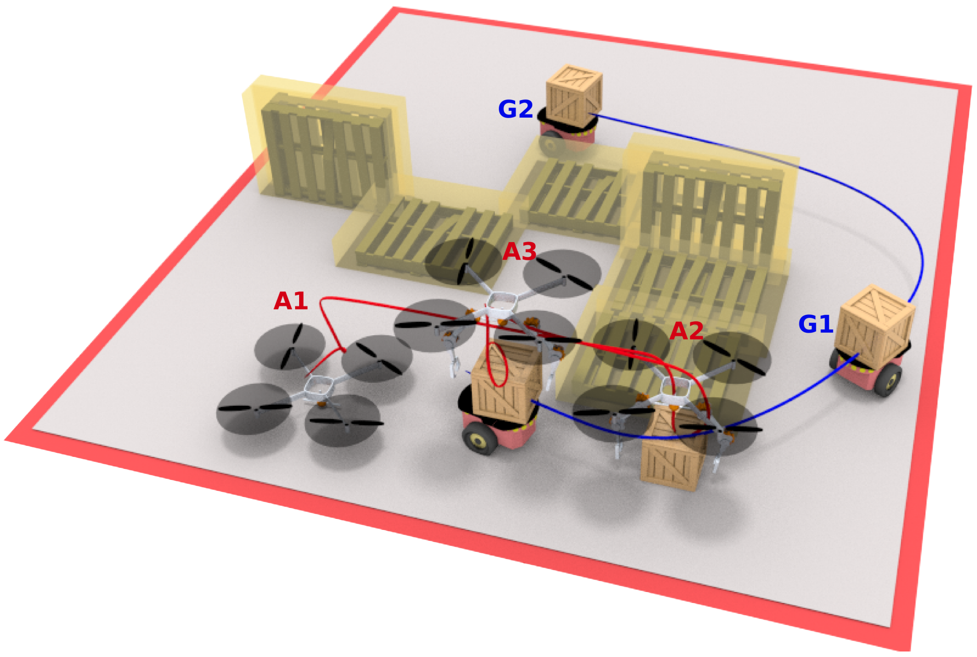


Figure 6.3: Experimental scenario: Designed solution where UAV and a single UGV are employed to deliver a parcel. UAV takes off (A1) and approaches the parcel. UAV grabs the parcel (A2), then flies just above UGV before releasing the parcel on the top of UGV (A3). The mobile robot finishes the mission by traveling (G1) and delivering the parcel (G2).

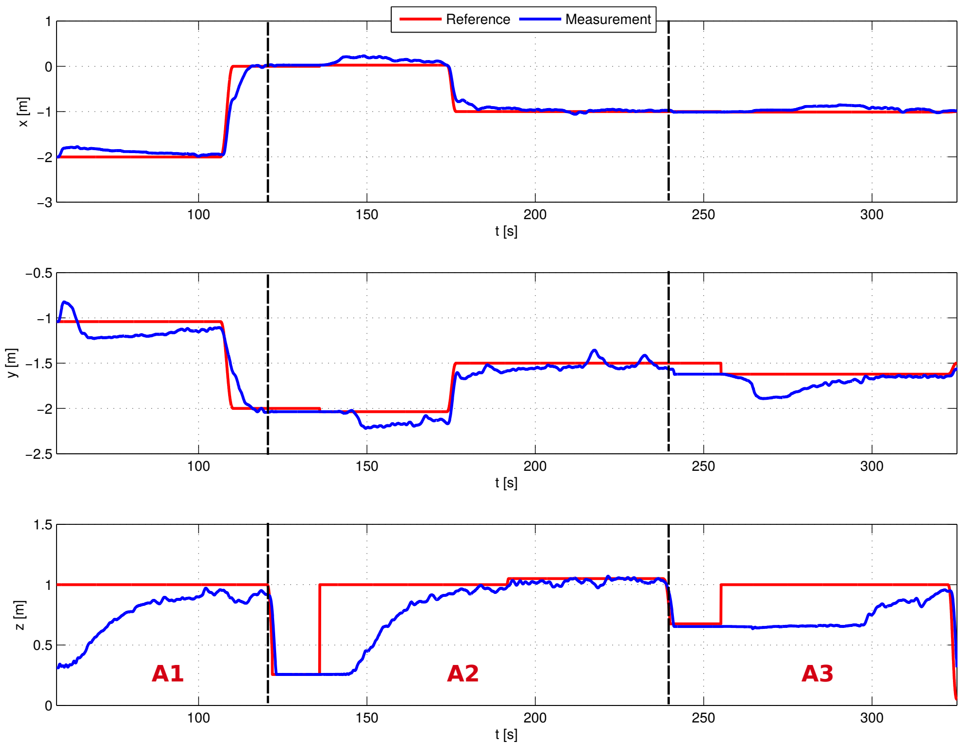


Figure 6.4: Quadrotor trajectories in the real-world experiment. Time segments A1-A3 correspond to the mission parts described in Figure 6.3.

mission are known a priori. The outputs of the sampling-based path planners are used for cost estimation in the planning process and as input for determining feasible obstacle-free trajectories. The approach was tested in a parcel transport scenario, first with the high-fidelity simulation environment Gazebo and then experimentally with a 3D Robotics quadcopter and two Pioneer 3DX mobile robots. The results show how the solutions of the planning procedure depend on the set criteria and on the environment itself. In subsequent work, we continue to work on a similar application scenario, with the goal to further increase energy savings based on more accurate task estimation that includes robot self-localization and mapping.

6.2 Decentralized Control of a UAV-UGV Motion-Symbiotic Team

The previous research focuses on manipulation missions using a dual-arm manipulator aboard a UAV to create an *Unmanned Aerial System* (UAS), formerly referred to as an MMUAV, to cooperate with UGVs in joint missions. However, most commercially available UAVs cannot be used for complex aerial manipulation tasks that normally require lifting heavy objects and long execution times. To overcome these limitations, in [100], instead of building UAVs with better capabilities, we propose the introduction of UGVs that support and accompany the UAS in complex aerial manipulation scenarios. In such a system of robots, the UGV provides the UAS with a safe landing site and transports it over long distances, saving valuable UAS battery life.

We propose a novel kind of aerial-ground system that focuses on symbiotic behavior that expands the movement capabilities of each vehicle. To this end, we have designed and constructed a *lightweight UGV* (L-UGV) suitable to work closely with a UAS, and specifically designed to be transported by the UAS. Thus, the two vehicles complement each other to form a symbiotic aerial-ground robotic system. The robots used are pictured in Figure 6.5.

To test the proposed system, we imagine a scenario where a team of robots is deployed to retrieve a parcel. The parcel is placed in various cluttered environments. The robots are instructed to negotiate an optimal solution (either in terms of energy consumption or mission duration) that involves a series of actions that allow the system to map the environment and retrieve the parcel.

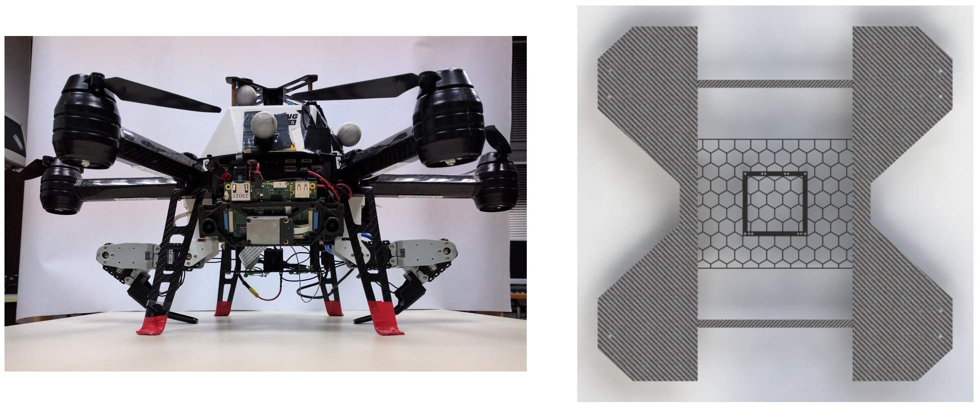


Figure 6.5: UAS and L-UGV used in the parcel delivery task.

6.2.1 Mission specification

The envisioned application scenario involves autonomous UASs and L-UGVs working cooperatively to transport a parcel between two locations in the environment. The inputs to the described planning procedure are a 3D occupancy map of the environment, the origin and destination locations of the parcel, and a local view of the mission tree describing the parcel transport scenario (mission).

The mission structure of the described scenario can be quite complex, depending on the number of obstacles in the environment. At this point, only the most important groups of tasks are outlined, since the mission tree is too extensive to be conveniently included in the thesis. As Figure 6.6 shows, at the root of the mission tree is the task *Transport parcel*, which is further decomposed into a set of subtasks *Cross the obstacle*, followed by *Pick up parcel* and another set of tasks *Cross the obstacle*, and finally *Deliver parcel*. The graph shown is for the general case of an environment with N obstacles on the way from the origin point to the location of the parcel. All subtasks are precedence-constrained by the node to their left thanks to the quality accumulation function *seq_sum_all* (*AND* operator with precedence constraints enforcing sequential execution of all subtasks) of the task *Transport parcel*.

Each *Cross the obstacle* task is divided into several options for crossing the obstacle, and can only be accomplished by performing one of them (the defined QAF is *q_max*, corresponding to the logical operator *XOR*). The decision is whether to tackle the obstacle together or have the UAS cross it alone. If the UAS were to cross one of the obstacles alone, all subsequent *UAS continue with L-UGV* tasks in the same mission would not be allowed to be executed until the UAS returns to the same obstacle where the L-UGV is waiting. To enable such behavior, we extended the mission representation by introducing another relation between tasks, the *excludes* relation. This relation simply lists tasks that cannot appear together in the same schedule, and such instances are discarded before

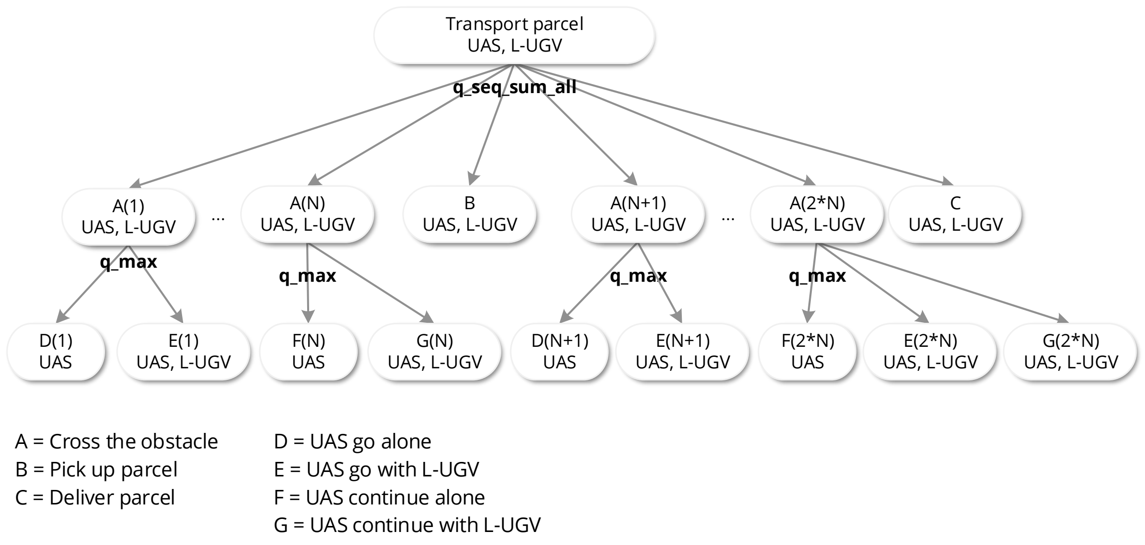


Figure 6.6: Mission decomposition of the parcel delivery task.

scheduling. For instance, the graph shown in Figure 6.6 contains the following excludes relations: $excludes(D(1), G(i))$, $\forall i \in (2, 2*N))$, where N denotes the number of obstacles. Similarly, excludes relations are defined between other tasks that do not semantically belong to the same execution sequence, $excludes(D(i), G(j))$, $\forall j \in (i+1, 2N-2i+2))$.

The proposed model has been proven in simulations to generate schedules with a large energy saving. All possible solutions produced from the mission specification are shown in Figure 6.7, where the estimated energy cost of each solution is illustrated with a bar next to it and the exact value in MJ. Figure 6.7 shows a symbolic representation of a mock-up arena with two obstacles between the starting position (far left) and the parcel position (far right). The path segment lengths are $10m$, $10m$, and $1.5m$, starting from the left. The average velocities of the robots are set to $0.13m/s$, and the energy expenditure is calculated using the power consumption for each robot-action pair as given in Table 6.3.

The first figure represents a baseline solution where the UAS performs the entire mission alone and its energy expenditure is estimated to be $0.194MJ$. As can be seen in Figure 6.7, the solution where the L-UGV is deployed on the first two segments in both directions but the UAS flies over the rightmost obstacle alone outperforms any other schedule (case (6)). Since the third segment is much shorter than the previous two, it is not worthwhile to cross the last obstacle with the L-UGV. An interesting alternative is the one in case (3), where the UAS uses the L-UGV to navigate the first two segments but returns to the starting point alone, which could be beneficial if the L-UGV needs to stay close to the target for further exploration.

The proposed system is experimentally tested on a real system consisting of a UAS and a specially designed lightweight UGV. The experimental setup and the obtained results

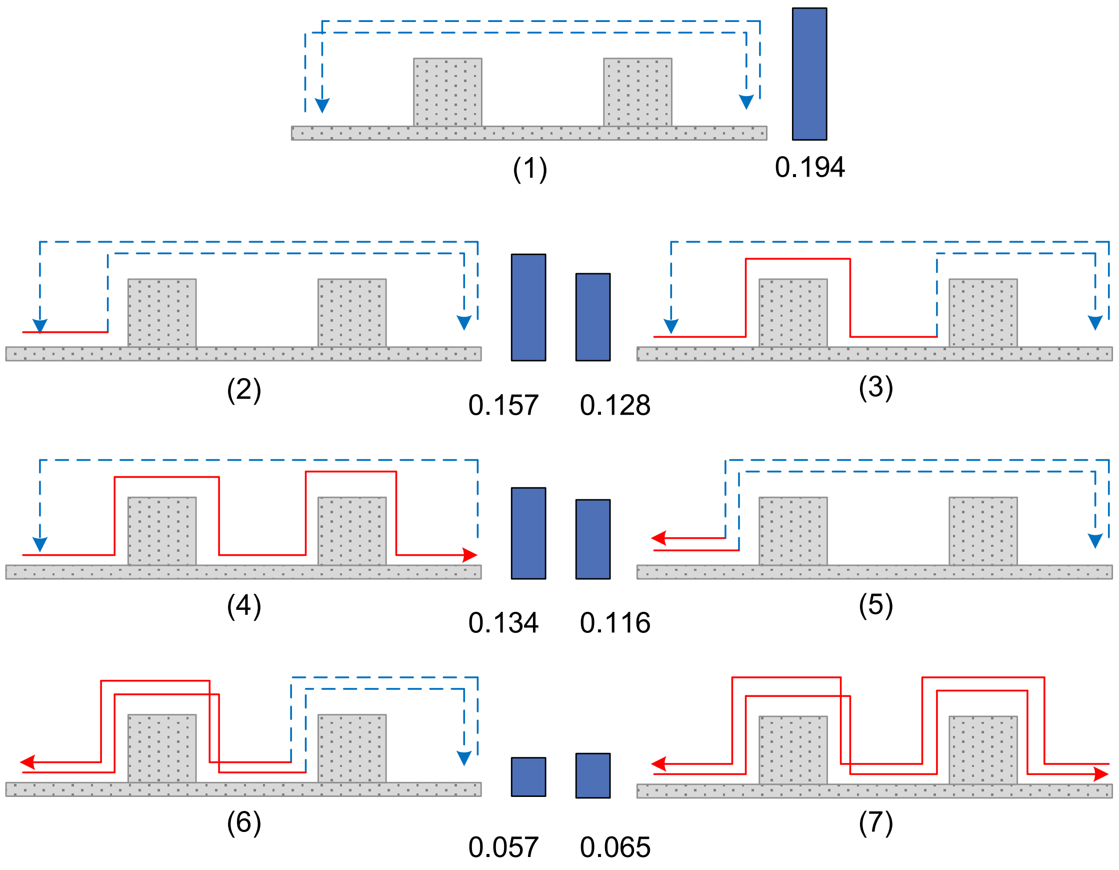


Figure 6.7: Different solutions obtained in the simulation for a parcel delivery mission. The red line represents paths UAS and UGV traverse together, while dashed blue lines denote paths taken by UAS alone.

are presented in the following sections.

6.2.2 Experimental setup

In the experiments, we recreated an arena with narrow corridors and obstacles along the way. As mentioned earlier, here the mapping and localization of the environment are performed autonomously, as opposed to having a static map and the use of an external localization system.

For mapping and localization, we use visual stereo odometry based on feature selection and tracking (SOFT) [101]. The algorithm uses the *VI-sensor*, which provides images from two cameras as well as IMU measurements. In addition, the algorithm provides a 3D map of the environment as *OctoMap*, in which the path and trajectory for the UAS are planned. However, in order to successfully guide the ground robots through the environment, the 3D map is projected onto the ground.

To achieve easier interaction with the mission planner, a set of behaviors (actions) for

Table 6.1: List of robot behaviors in symbiotic parcel delivery mission.

Behavior	**Description**
Takeoff	The UAS goes to the desired height above its current position
Land	The UAS lands below its current position
Move to desired position	This behavior is common for both robots (UAS and L-UGV). The robot moves to the requested position and holds that position until a new request is received.
Hold position	This behavior is common for all robots. The robot holds position until a request for a different action is received.
Grab L-UGV	Enables visual tracking of the L-UGV, descends and grabs the L-UGV upon successful detection
Release L-UGV	Executes preplanned dual manipulator motion to release the L-UGV and set the manipulator in the soft home position
Grab parcel	Enables visual tracking of the parcel, descends and grabs the parcel upon successful detection
Release parcel	Executes preplanned dual manipulator motion to release the parcel and set the manipulator in soft home position
Land on the L-UGV with the parcel	Enables visual tracking of the L-UGV while parcel is acquired, descends to the L-UGV, releases the parcel and further descends in order to grab the L-UGV

the UAS was developed as described in Table 6.1. The described behaviors had to be tailored to our experimental setup. The dimensions of both the parcel and the L-UGV had to be considered for the pickup and delivery behaviors. Landing on the L-UGV while a parcel is being transported is a two-stage behavior. In the first stage, the UAS descends over the L-UGV and prepares to release the parcel. Due to ground effect and other possible interference, the UAS must wait for the right moment to drop the parcel. When the conditions for approaching the drop point are met, the UAS releases the parcel, rapidly descends further and grabs the L-UGV. The action is completed with the landing.

6.2.3 Experimental results

During the experimental verification, we performed several different missions to analyze the efficiency of the planning method. To perform the missions, we used an L-UGV - UAS team described in the previous sections. As a baseline mission for the energy consumption analysis, we conducted the first parcel delivery mission using only the UAS. At the beginning of each mission, we conduct a reconnaissance of an unknown area to map the environment and locate areas of interest. The goal of all missions is to reach the target identified in the reconnaissance part and perform the required actions, e.g., inspect the area, pick up the parcel, etc.

In the first environment, we performed three missions of the parcel delivery task, differing only in the robots used and the task decomposition. These experiments were designed to illustrate the energy saving capabilities of our system. Next, we experimented with different environmental configurations to demonstrate the robustness of the proposed method in variable environments.

We calculate the consumed energy by integrating the power required for each action over the time taken. Each action requires different amounts of energy due to different combinations of total mass, as shown in Table 6.2.

All the tasks that the system must perform are composed of the four basic actions: Takeoff (A1), Land (A2), Fly (A3), and Drive (A4). Based on the characteristics of the drone (i.e., total battery capacity $Q_{p,uav} = 8000mAh$, voltage $V_{avg,uav} = 15.8V$, and flight time $t_f \approx 10min$), along with the measured values of the L-UGV, we compute the power requirements for each action-class pair given in Table 6.3.

Energy conservation capability

The first environment configuration is shown in Figure 6.8. The task of the robotic system was to navigate through the maze, pick up the parcel at the end of the maze, and deliver it to the start position. The optimization criterion was to minimize the total energy

Table 6.2: System classification with regard to mass.

		C1	**C2**	**C3**	**C4**
UAS	$2.7kg$	X	X	X	X
L-UGV	$0.4kg$		X		X
parcel	$0.1kg$			X	X
		$2.7kg$	$3.1kg$	$2.8kg$	$3.2kg$

consumption. The first step for all missions was the exploration process during which the map of the environment was constructed.

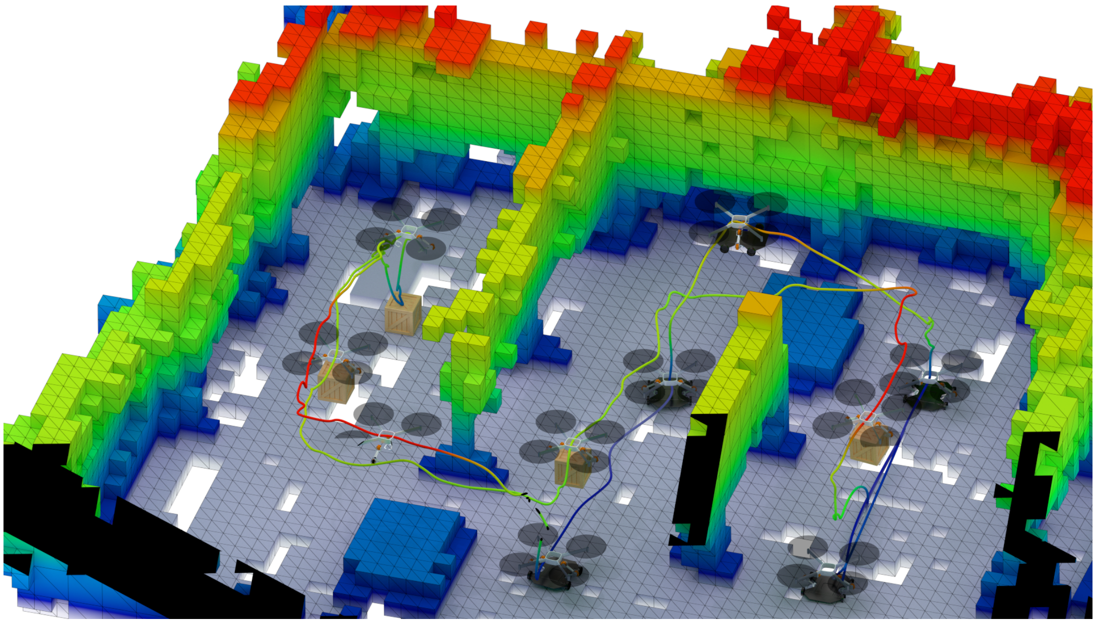


Figure 6.8: Experimental result for the parcel delivery task where the L-UGV is deployed in one direction only (Mission 2). L-UGV drives the UAS to the first obstacle, UAS takes off with the L-UGV, crosses the obstacle and then lands. L-UGV continues with the UAS to the next obstacle, after which the UAS takes off without the L-UGV and flies on alone to pick up and deliver the parcel.

Mission 1: The baseline for the first three experiments is a mission with only UAS. In this mission, we get the base energy consumption that we want to reduce in other missions using auxiliary robots. The UAS executes a trajectory to position itself above the parcel, which it then picks up and executes a return trajectory.

Mission 2: In this mission, we deploy the L-UGV. At the beginning of the mission, the UAS rests on top of the L-UGV and holds on to it. The UAS releases the L-UGV before the second obstacle and continues the mission alone, as happened in Mission 1. As shown in Table 6.5, the energy savings from using L-UGV in one direction only is significant and

Table 6.3: Power needed to execute each action with regard to overall mass.

	C1	**C2**	**C3**	**C4**
A1	579W	711W	616W	731W
A2	566W	696W	602W	729W
A3	570W	701W	606W	744W
A4	N/A	N/A	3.3W	3.3W

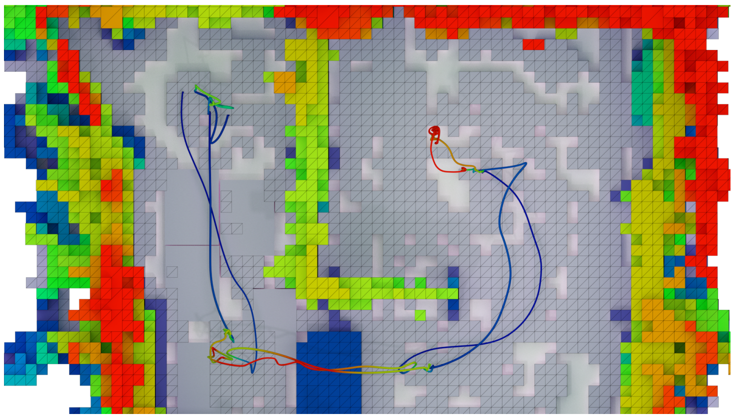
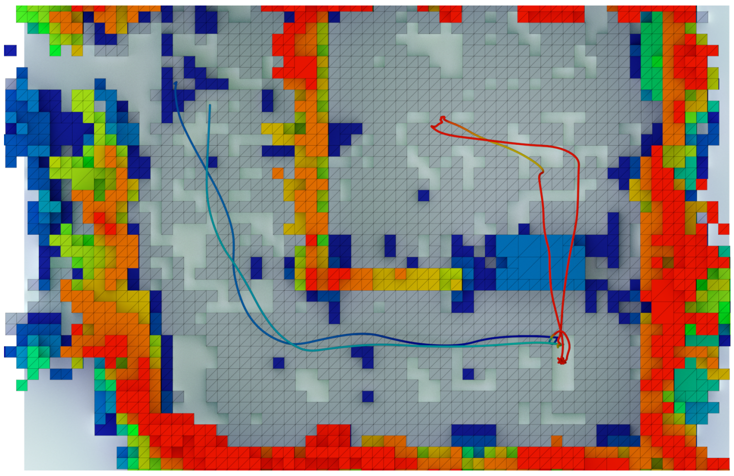
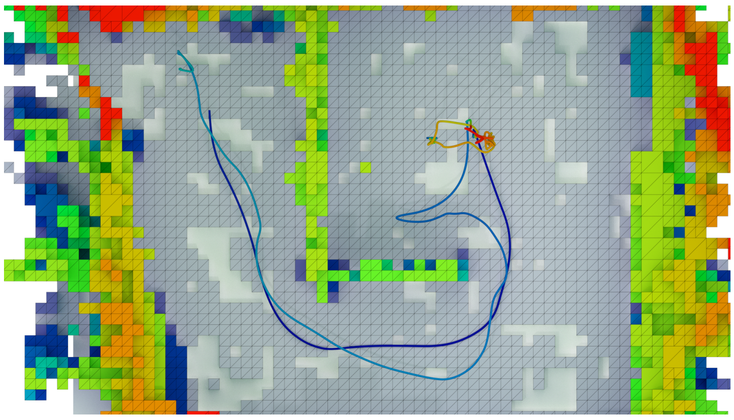
	Mission 4: L-UGV travels with UAS to obstacle, UAS takes off with L-UGV, flies over obstacle and lands behind it. L-UGV continues with UAS to target position, where it detaches from L-UGV, takes off, and performs inspection task. Then the UAS lands on the L-UGV and they continue in the same manner to return to the start position.
	Mission 5: L-UGV drives UAS to obstacle, at which point UAS detaches from L-UGV, takes off, and continues autonomously to target position. UAS performs inspection task, flies back to L-UGV and lands on it. L-UGV returns with the UAS to the start position.
	Mission 6: L-UGV moves UAS to position near the parcel. UAS takes off, positions itself over the parcel, grabs it, and returns to L-UGV. UAS lands on L-UGV while holding onto the parcel, and L-UGV drives to start position with parcel and UAS.

Table 6.4: Experimental results for parcel delivery and inspection tasks on variable environments.

is 36.7%.

Mission 3: To further exploit the energy saving capabilities of the system, we enabled the use of L-UGV during the return to the start position with the parcel. To do this, we had to provide for an action of landing on the L-UGV while carrying the parcel, which proved to be non-trivial. The resulting schedule is similar to the previous one, but differs in the utilization of the L-UGV in both directions, as shown in Figure 6.7, case (6). With this schedule, we were able to improve the energy savings by an additional 44.7% of base energy, for a total savings of 81.4%.

Table 6.5: Energy expended in different experimental setups

	Energy (J)	**Energy saving (%)**
Mission 1	542.150	baseline
Mission 2	343.118	36.7
Mission 3	101.098	81.4

6.2.4 Robustness to variable environment

To demonstrate effectiveness of our method in variable environments, we devised three additional environment setups, as shown in Figure 6.4. Missions 4 and 5 are not focused on parcel delivery, but rather on inspection (exploration) of the area of interest. Finally, in Mission 6, we completed the same objective as in Mission 3. The execution of the proposed scenario in selected environments is shown in video footage included in the playlist at [102].

From a high-level mission planning perspective, we have developed and verified a decentralized hierarchical planning method able to construct and coordinate, in real-time, feasible team plans for any given map of the environment. Planning is modular and able to cope with teams consisting of any number of UASs and UGVs. Our previous work is augmented to allow for mission representation with arbitrary number and arrangement of obstacles detected from the start point to the parcel.

In our following work, we focus on developing solutions to more complex problems from the perspective of task allocation. Therefore, the goal is to replace the basic greedy task allocation procedure (that assigns actions to robots with best outcomes, without considering other allocations) with one that finds solutions closer to the optimum.

6.3 Cooperative Aerial-Ground Multi-Robot System for Automated Construction Tasks

In the following, we study a cooperative aerial-ground robot team and its application to the task of automated construction as described in [103]. We propose a solution for planning and coordinating the construction of a wall with a predefined structure for a heterogeneous system consisting of UGV and up to three UAVs. The wall consists of bricks with different weights and sizes, some of which have to be transported by several robots simultaneously. To this end, we use a hierarchical task representation to specify the relationships between the subtasks of the mission and employ an effective planning and coordination mechanism inspired by GPGP. Finally, we evaluate the method's performance under different optimization criteria and validate the solution in the realistic Gazebo simulation environment.

The approach presented here involves online high-level task planning (decomposition, allocation, and scheduling) using TÆMS for task representation. The planner used in this work is based on GPGP framework defined in Section 3.2.1, with multiple improvements defined in this Section.

In this work, we extend the capabilities of the framework in terms of task allocation by implementing a market-based protocol for the problem class XD[ST-MR-TA] (cross-schedule dependencies, single-task robots, multi-robot tasks, time-extended assignment) [38]. Consequently, we can handle complex scheduling problems in real-time without compromising the optimality of the obtained solutions. Moreover, the protocol includes temporal and precedence constraints, which is not the norm in the state-of-the-art literature. We apply the method to the problem of automated construction, which perfectly highlights the planner's ability to deal with combinatorial optimization problems of heterogeneous multi-robot systems in complex missions.

6.3.1 Problem description

The *Mohammed Bin Zayed International Robotics Challenge* (MBZIRC)[†] aims to foster innovation and research at the highest level in emerging topics by providing a demanding set of robotics challenges that require robots to operate more autonomously in dynamic, unstructured environments while cooperating and interacting with each other. The technological challenges addressed in the MBZIRC 2020 Challenges include fast autonomous navigation in semi-unstructured, complex, dynamic environments, with reduced visibility (e.g., smoke) and minimal prior knowledge, robust perception and tracking of dynamic

[†]http://mbzirc.com

objects in 3D, obstacle detection and avoidance, GPS denied navigation in indoor-outdoor environments, physical interactions, complex mobile manipulations, and air-ground collaboration [104]. In this section, we address the problem of automated construction, where a team of aerial and ground robots must collaborate to autonomously locate, pick, transport, and assemble different types of brick-like objects to build predefined structures. In particular, we are interested in the decision-making and coordination process that optimizes the wall-building task given multi-objective criteria.

The task is given as follows. Red, green, blue, and orange bricks with a length of $0.3m$ to $1.8m$ are separated by color and assembled in randomly arranged piles before the start of the challenge. Blue bricks may only be collected by the UAVs, while orange bricks must be carried by two or more UAVs simultaneously due to their size and weight. Both the UAVs and the UGV can assemble red and green bricks, but those built by the UAVs are scored higher in the challenge. The shape of the wall structure is the only input into the system. Teams have 30 minutes to complete the challenge, and the speed and percentage of completion determine the score. The exact parameters of the challenge and the properties of the bricks are further specified in [104].

This challenge was specifically designed to emphasize the need for collaboration between different types of robots. UAVs are maneuverable, have larger operational range, and are superior in numbers, which means they can build more in less time. However, due to their limited battery life, it is imperative to use the UGV for low-scoring bricks and those closer to the ground.

The focus of this work is to implement a high-level decentralized mission planning and coordination algorithm to oversee the task of construction. All available robots are expected to perform to their full potential and execute assigned tasks in parallel to maximize the challenge score and complete the task within the given time.

6.3.2 Mission specification

We decompose the wall construction task using the TÆMS hierarchical task structure such that the immediate subtasks of the root task are the transportation and assembly of the individual bricks. The TÆMS tree is defined for three types of agents, "UGV", "UAV" and "UAVx2", where the last label denotes an agent consisting of two UAVs that can cooperate in transporting the large orange brick. The robot behaviors (actions) used in this mission are $GP(b_i)$ - go to the pile and locate the brick b_i, $PU(b_i)$ - pick up the brick from the pile, $GW(b_i)$ - go to the wall, and $PD(b_i)$ - place the brick at the specified location in the wall, where i denotes the brick identifier. A task node that combines all the actions necessary to transport a single brick b_i is denoted by $TB(b_i)$.

In this model, two relations between nodes define how the quality, duration, and cost of child nodes affect the parent node – *q_sum_all*, which corresponds to logical AND, *q_seq_sum_all*, which is AND with strictly sequential execution. *Quality* of each task represents the number of points obtained for the successful assembly of each building block. *Duration* of a task is estimated based on the length of the desired path and the average movement speed of the robot. In addition, for the actions $PU(b_i)$ and $PD(b_i)$, we add a fixed estimated duration of the grasp and release behavior. Finally, we determine the *cost* (relating to energy expenditure) by multiplying the duration by the cost per time based on the complexity of the task, the type of robot performing it, and the size of the associated brick.

If a task cannot be completed due to kinematic constraints of the robot (e.g., the desired placement of the brick is unreachable by the robotic arm of the UGV), the quality of such a task is set to zero, while duration and cost are set to a large positive number to exclude it from planning.

Relationships between other action and task nodes are modeled using interrelationship TÆMS elements. *Enables* relations require one task to finish before starting another, and are used to determine the order of bricks within the wall structure (precedence relation). The action of picking up a brick ($PU(b_i)$) precedes the action of going to the pile to pick up another brick ($GP(b_j)$).

This introduces an offset when transporting adjacent bricks, and helps avoid situations where multiple robots simultaneously place bricks in close proximity. The offset is introduced at the beginning of the tasks because it is more efficient for the UAVs to wait on the ground than in the air while transporting a brick. *Disables* relations are used to further prioritize the order of brick placements to ensure schedule feasibility.

6.3.3 Resolving simple redundancy

One of the most important steps in the GPGP coordination process is resolving simple redundancies between tasks (as defined in Section 3.2.1. Redundancy is a situation where one or more robots in the set denoted by P_{sr} can perform a single task. The resolution procedure starts by selecting one of the robots as the referee. The selected robot then collects task scores from all the other robots, selects the best one, and communicates the results. Robots that are not selected as the best discard the task from their plans.

The first part of the criterion function for selecting the best robot to perform a redundant task j is defined as follows:

$$r_{Q,i}(j) = \frac{Q_i(j) - Q_{min}(j)}{Q_{max}(j) - Q_{min}(j)}, \tag{6.1}$$

$$r_{D,i}(j) = \frac{D_{max}(j) - D_i(j)}{D_{max}(j) - D_{min}(j)}, \tag{6.2}$$

$$r_{C,i}(j) = \frac{C_{max}(j) - C_i(j)}{C_{max}(j) - C_{min}(j)}, \tag{6.3}$$

$$R_i(j) = \alpha \cdot r_{Q,i}(j) + \beta \cdot r_{D,i}(j) + \gamma \cdot r_{C,i}(j), \tag{6.4}$$

where $Q_i(j)$ is the quality of task j assessed by robot $i \in \mathrm{P}_{sr}$, $Q_{min}(j) = min_{i\in\mathrm{P}_{sr}}Q_i(j)$ and $Q_{max}(j) = max_{i\in\mathrm{P}_{sr}}Q_i(j)$. Analogous definitions exist for $D_i(j)$, $D_{min}(j)$, and $D_{max}(j)$, which represent duration assessment, and $C_i(j)$, $C_{min}(j)$, and $C_{max}(j)$ for costs. The parameters α, β and γ are user-defined positive real constants, $\alpha + \beta + \gamma = 1$, which give different importance to various criteria.

The problem occurs when redundant tasks or robots are very similar. In this case, the task scores are almost identical, but usually, one robot has a marginally better outcome and is chosen to perform all redundant tasks. Therefore, other robots are not used, and the tasks cannot be executed in parallel. Instead, we use additional market-based criteria function to allocate the redundant tasks better, as described in the following paragraph.

Based on the initial estimates of the duration and relationships between tasks, a simple market-based task allocation function creates an assignment scheme that aims to minimize the total mission duration. The output of the function is a list S of pairs in the form $(task,\ robot\ assigned\ to\ the\ task)$. Such an assignment is a preferred solution for redundancy (r_2), but the task assessments of the individual robots (r_1) are still used to cover the cases where one robot has a significantly better execution result. The overall rating of the robot i for the given task j is:

$$r_{1,i}(j) = \frac{R_i(j) - R_{min}(j)}{R_{max}(j) - R_{min}(j)}, \tag{6.5}$$

$$r_{2,i}(j) = \begin{cases} 1, & \text{if } (j,i) \in S, \\ 0, & \text{otherwise}, \end{cases} \tag{6.6}$$

$$R_{total,i}(j) = \delta \cdot r_{1,i}(j) + (1 - \delta) \cdot r_{2,i}(j), \tag{6.7}$$

where $R_i(j)$ is a rating of the task j based on quality, duration, and cost assessment of the robot $i \in \mathrm{P}_{sr}$, as defined in (Eq. 6.4), $R_{min}(j) = min_{i\in\mathrm{P}_{sr}}R_i(j)$, $R_{max}(j) = max_{i\in\mathrm{P}_{sr}}R_i(j)$, and δ is a user-defined positive constant, $\delta \in [0.5, 1)$, that gives different importance to the two parts of the criteria function. Note that for $\delta < 0.5$, rating r_1 would not affect the total rating because $r_{1,i}(j) \leq r_{2,i}(j),\ \forall i \in \mathrm{P}_{sr}$ and $\sum_{i\in\mathrm{P}_{sr}} r_{2,i}(j) = 1$. In other words, for each task j, there would be only one robot for which $r_{2,i} = 1$ and that robot would have better total rating than all others, regardless of the value of r_1.

6.3.4 Resolving complex redundancy

Complex redundancy is an extension of the TÆMS model to include a so-called local Quality Accumulation Function (QAF). It represents the situation where a task has multiple subtasks that need to be executed simultaneously, and therefore a one-to-one mapping between subtasks and robots is required. Such a situation occurs in the case of the task of joint transportation of orange brick. Two robots have to schedule their part of the task at the same time and execute it jointly. Globally, both subtasks must be executed to complete the task, while locally only one of these tasks should be scheduled for each robot. Therefore, tasks are modeled as complexly redundant by having QAF of their parent task as *q_sum* (logical OR) or *q_sum_all* (logical AND) and QAF_{local} as *q_max* (logical XOR). We use the term redundancy because the effect is similar to the simply redundant tasks, but its resolution must be handled differently. Instead of multiple robots competing for a single task, as is the case in the resolution of simple redundancy, m robots compete for n tasks, $m \geq n$, so that each robot schedules only one subtask and all subtasks are assigned.

The problem of assigning only one robot to each task in a way that optimizes overall quality, duration, and cost is modeled as a generalized assignment problem [105]:

$$\max \quad \sum_{i \in \mathrm{P}_{cr}} \sum_{j \in T_{cr}} R_i(j) \cdot x_{ij}, \tag{6.8}$$

$$\text{subject to} \quad \sum_{i \in \mathrm{P}_{cr}} x_{ij} = 1, \ \forall j \in T_{cr}, \tag{6.9}$$

$$\sum_{j \in T_{cr}} x_{ij} = 1, \ \forall i \in \mathrm{P}_{cr}, \tag{6.10}$$

where P_{cr} is a set of robots capable of scheduling a complexly redundant task, T_{cr} is a set of subtasks of the complex redundant task, $R_i(j)$ is the evaluation of subtask j based on the evaluation of robot i defined in equation (6.4), and $x_{ij} \in \{0, 1\}$ is a binary decision variable for the assignment of robot i to subtask j. The equations (6.9) and (6.10) ensure that each subtask is assigned to only one robot and that each robot performs only one subtask, respectively. The described optimization problem is solved with a custom implementation of the branch-and-bound algorithm [106].

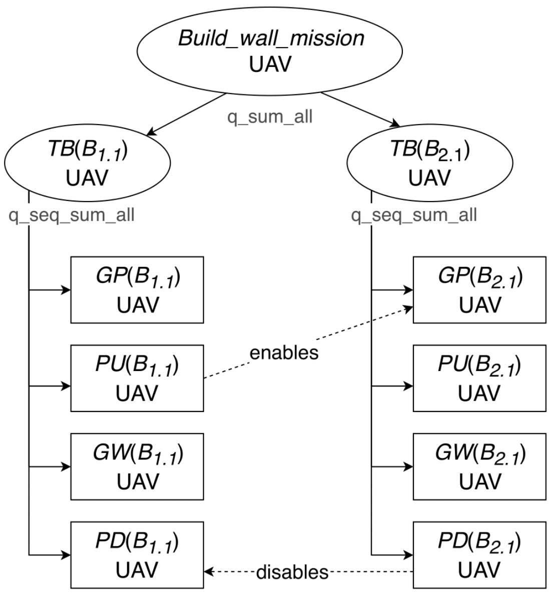


Figure 6.9: Task structure of the example mission with two bricks.

6.3.5 Resources

Each set of action nodes is associated with its parent task with a *q_seq_sum_all* function, which means that they must be executed in a strictly specified order. However, this does not prevent a robot from scheduling and executing other tasks in the meantime.

Consider a system with only one UAV and two blue bricks to be stacked on top of each other. TÆMS tree structure for such a mission is shown in Figure 6.9. The bottom brick is labeled $B_{1.1}$, while the top one is labeled $B_{2.1}$. The expected execution schedule for the given mission is as follows:

$$GP(B_{1.1}),\ PU(B_{1.1}),\ GW(B_{1.1}),\ PD(B_{1.1}),$$
$$GP(B_{2.1}),\ PU(B_{2.1}),\ GW(B_{2.1}),\ PD(B_{2.1}).$$

However, successful execution of action $PU(B_{1.1})$ enables action $GP(B_{2.1})$ and final execution schedule is in fact:

$$GP(B_{1.1}),\ PU(B_{1.1}),\ GP(B_{2.1}),\ PU(B_{2.1}),$$
$$GW(B_{1.1}),\ GW(B_{2.1}),\ PD(B_{1.1}),\ PD(B_{2.1}).$$

The scheduling algorithm is intentionally unaware of the robot's specific abilities, so it is suitable for different applications. The best-found schedule is the best solution for the given problem because it minimizes the total mission duration, even though UAVs cannot transport multiple bricks simultaneously. To model this constraint, virtual resources are used in the extended model of the mission. Each robot has a resource that represents a slot

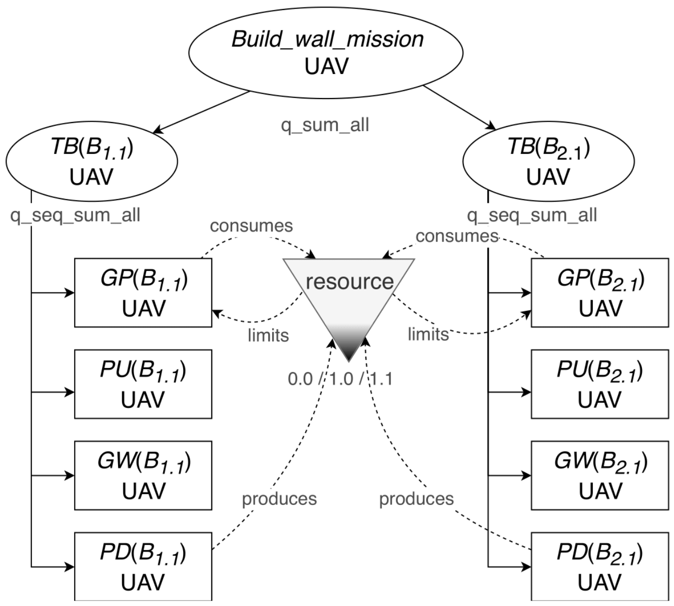


Figure 6.10: Task structure of the example mission extended with resources. *Enables* relations are omitted for clarity.

for carrying a brick. The initial state of the resource is 1, while its lower and upper bounds are 0 and 1.1, respectively. Every time a robot performs an action of type $GP(b_i)$, one unit of the resource is consumed, making it insufficient. Insufficient resources automatically disable all other $GP(b_j)$, $(i \neq j)$ actions for that robot until their state is restored by executing the $PD(b_i)$ action. The improved model of the TÆMS tree structure is shown in Figure 6.10.

6.3.6 Testbed description

We devised the environment for the second challenge of MBZIRC 2020 in Gazebo simulator (Figure 6.11). Using the ROS interface, we can realistically simulate planning, coordination, and execution aspects of the proposed solution for wall building mission.

We simulate physically accurate models of multicopters for UAV agents and a Gazebo model of Husky by Clearpath Robotics with mounted Schunk Powerball LWA 4P robotic arm for the UGV agent. Both models have standard sensors such as an inertial measurement unit (IMU) and a generic pose sensor that provide the true position and orientation of the robots. UAVs use a simple position PID controller for movement, while UGV uses a navigation algorithm provided by *move_base* ROS package. Mid-air collisions are prevented by commanding each UAV to a different predefined altitude.

Since this work focuses on planning and coordination, the robots do not have end effectors on board to interact with the bricks. In the simulation, the bricks are teleported using Gazebo's *spawn* and *delete* services. The locations of all four brick stacks and the wall origin are known before the mission begins. In the real world, challengers should

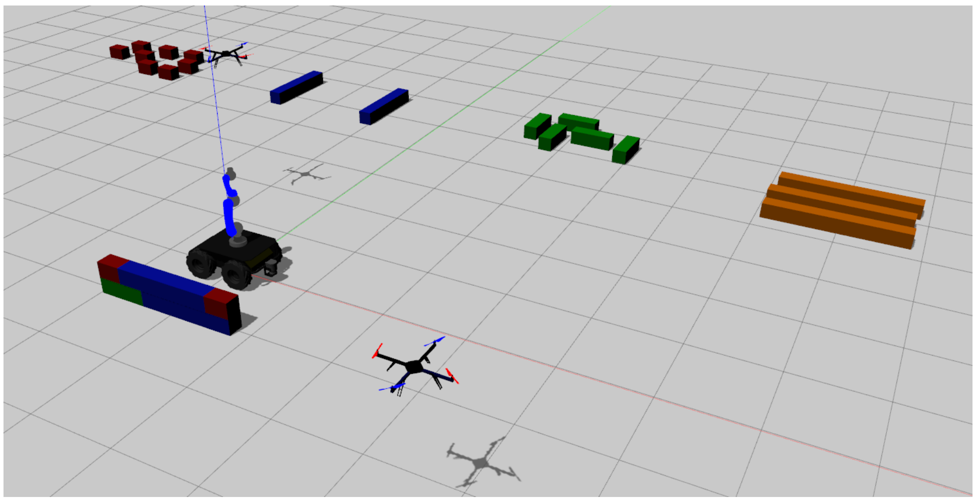


Figure 6.11: Gazebo simulation environment with robots.

obtain this information by scouting the arena with UAVs at the start of the trial.

6.3.7 Simulation results

To test the system described in this section, we deploy the team consisting of two UAVs and one UGV on a task specified as in Figure 6.12. We performed the simulated mission using several different criteria setups, three of which we analyze here. The cost of execution per unit time is held constant for UAVs and UGV in all scenarios.

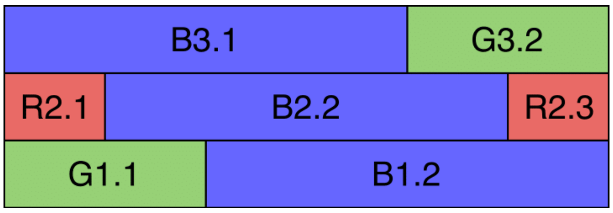


Figure 6.12: Specified wall structure for the first mission.

In the first scenario, we define the overall mission quality as the most weighty component of the planning criteria, while other components play a less important role, $\alpha = 0.5, \beta = 0.35, \gamma = 0.15$. As expected, only UAVs participate in the construction process since the bricks thus assembled yield more points. Our task allocation method alternately assigns tasks to the two UAVs to achieve parallel execution and shorten the overall mission duration. The timing diagram of the tasks executed by each robot is shown in Figure 6.13. Colored segments divide each task $TB(b_i)$ into its actions, where blue corresponds to action GP, red to PU, yellow to GW, and green to PD. Dashed arrows represent *precedence* relationships that affect the final schedule.

In the second scenario, the cost component of the criteria $\alpha = 0.35, \beta = 0.15, \gamma = 0.50$ is given the greatest weight. Since the UGV has a much lower cost per unit time than the

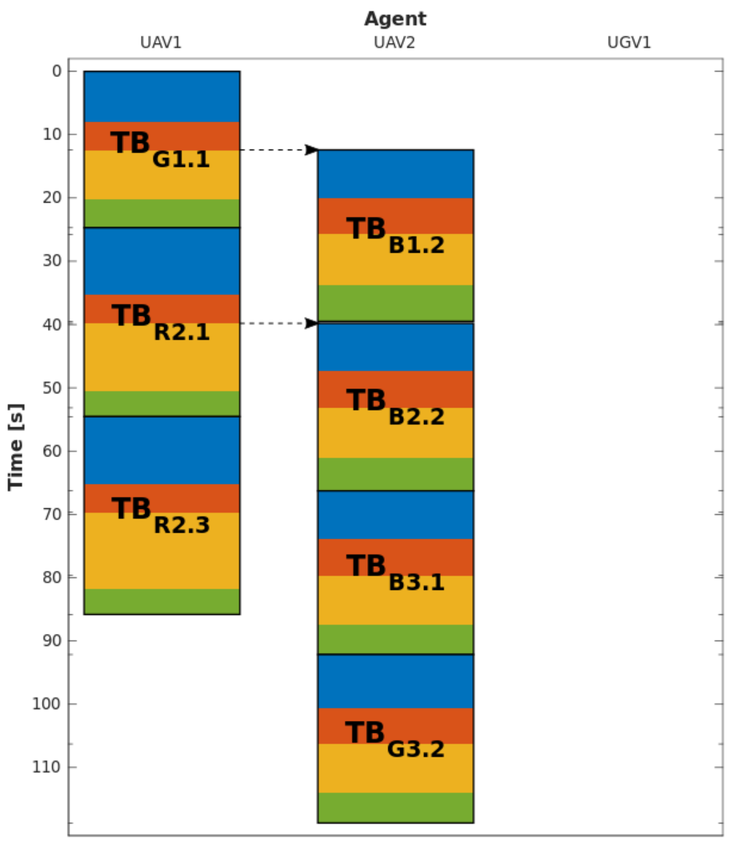


Figure 6.13: Execution schedule of the first mission for the criteria defined as $\alpha = 0.5, \beta = 0.35, \gamma = 0.15$.

UAVs, tasks related to green bricks are assigned to the UGV. We defined that red bricks provide 100% more points when assembled by UAVs, as opposed to 40% for the green bricks. Since mission quality still matters to some extent, a better result is obtained when UAVs assemble red bricks. The execution schedule for this scenario is shown in Figure 6.14.

Figure 6.15 shows the execution schedule of the third scenario, which is entirely focused on minimizing the total cost of the mission. The criterion is defined as $\alpha = 0, \beta = 0, \gamma = 1$. In this case, tasks related to red bricks are also assigned to the UGV since their increased score potential has no impact on the mission outcome. Blue bricks will continue to be assembled by the UAVs as defined in the challenge description.

Finally, we add a third UAV and deploy the team of robots on a more complex mission specified as in Figure 6.16. To reduce the mission duration but still include the UGV in the execution, we use the same criteria as in the second scenario of the first mission, $\alpha = 0.35, \beta = 0.15, \gamma = 0.5$.

The generated execution schedule for the final mission is shown in Figure 6.17. As in the previous scenarios, the tasks are evenly distributed among the robots to shorten the overall mission duration and reduce the total cost while maximizing the score. The results

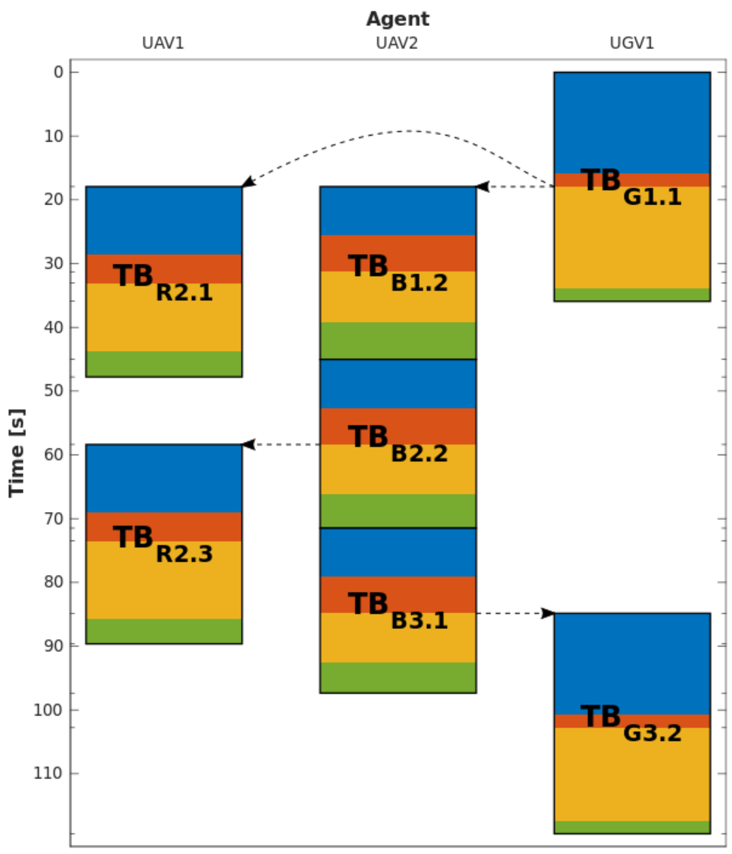


Figure 6.14: Execution schedule of the first mission for the criteria defined as $\alpha = 0.35, \beta = 0.15, \gamma = 0.50$.

confirm that the proposed method can generate optimized high-level plans for different robot and criteria setups and satisfies the current challenge requirements.

We ran 20 additional simulations for problems of similar dimensionality and compared performance with solutions obtained using the Gurobi Optimizer [44]. Gurobi Optimizer uses exact mathematical methods to solve Mixed-Integer Linear Programming (MILP) problems and can provide an optimal solution for the given mission set. We also compare the generated schedules with the state-of-the-art iterated auction-based approach with a single central agent acting as an auctioneer [55]. We tested each of the 20 mission scenarios on a set of five criteria specifications and plotted the average value and standard deviation of the objective function with respect to the optimal solution. The results of the experiments are shown in Table 6.6.

The results show that, on average, the proposed method is within 12% of the optimum for most of the tested criteria specifications, except for the criteria that are entirely focused on minimizing the mission makespan. Makespan is defined as the elapsed time between the start and finish of a sequence of tasks in a group of robots. When resolving simple redundancy, tasks are always assigned to faster robots (e.g., UAVs), which means that robots with significantly worse duration estimation (e.g., UGV) do not participate in

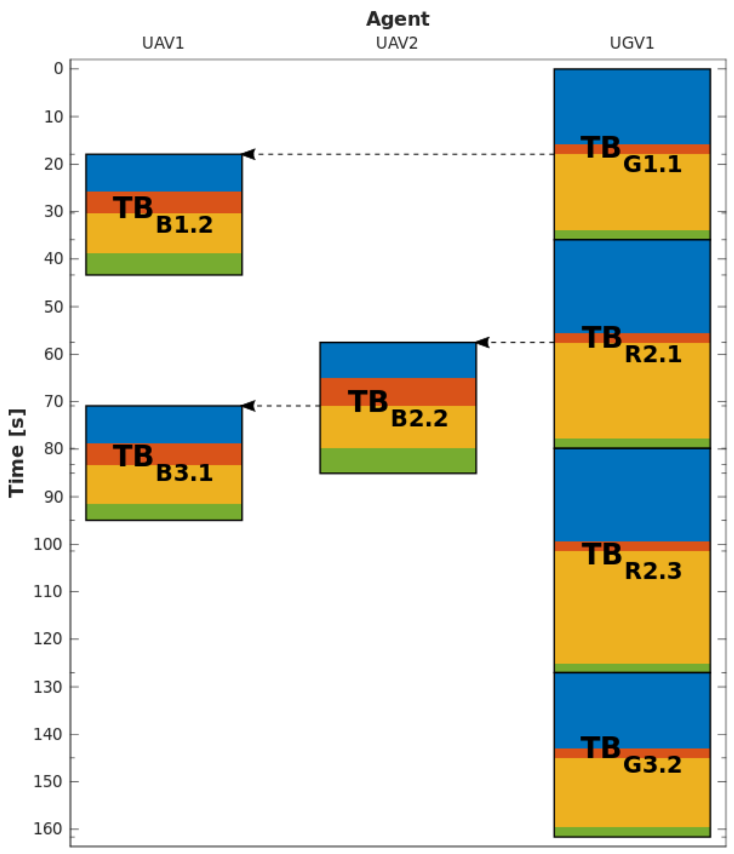


Figure 6.15: Execution schedule of the first mission for the criteria defined as $\alpha = 0, \beta = 0, \gamma = 1$.

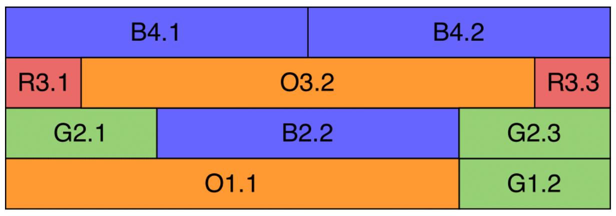


Figure 6.16: Specified wall structure for the second mission.

mission execution, thus increasing the overall duration.

The auction-based approach performs slightly better in terms of the achieved optimality gap. However, our approach is fully decentralized and can handle the scheduling and simultaneous execution of multi-robot tasks, e.g., transporting orange bricks. Another major advantage of the proposed method is that it is mission agnostic and only requires a hierarchical task structure, unlike Gurobi Optimizer, which requires an exact mathematical formulation. Moreover, the mathematical methods used by Gurobi often do not provide a valid solution when applied to more complex problems.

An example of two UAVs working together to assemble the large orange brick is also examined. Due to the complexity of this task, other tasks are temporarily suspended during its execution to avoid collisions and various interference. Therefore, precedence

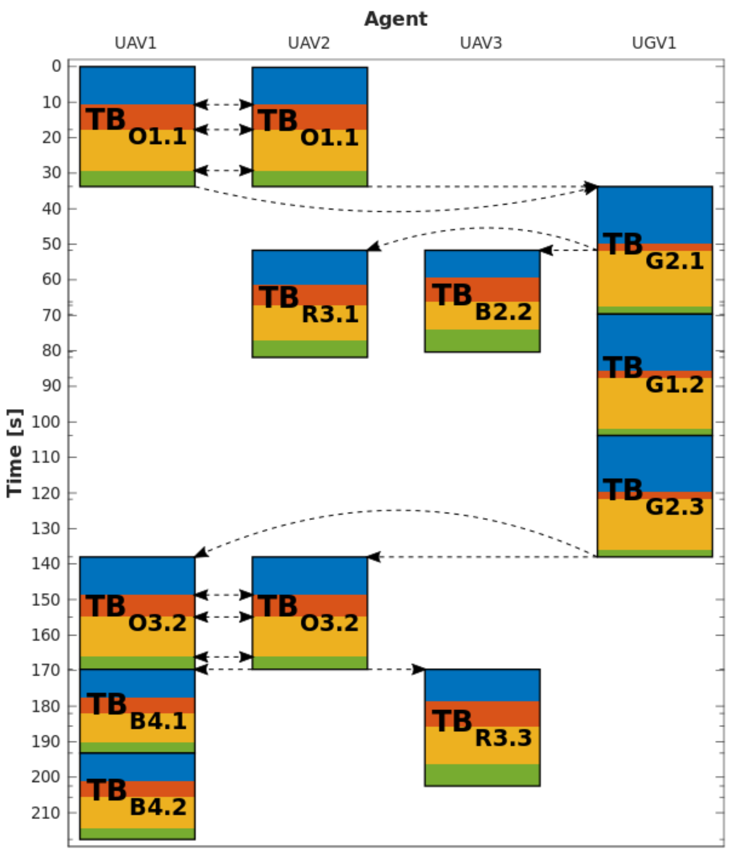


Figure 6.17: Execution schedule of the second mission.

Table 6.6: Optimality gaps of the proposed solution and auction based approach compared to Gurobi Optimizer. Lower values are better.

Criteria		Proposed solution	Auction
$\alpha = 0.5,\ \beta = 0.35,\ \gamma = 0.15$	μ	11.15 %	8.18 %
	σ	6.26 %	6.00 %
$\alpha = 0.35,\ \beta = 0.15,\ \gamma = 0.5$	μ	6.43 %	8.10 %
	σ	10.96 %	3.82 %
$\alpha = 1,\ \beta = 0,\ \gamma = 0$	μ	0 %	0 %
	σ	0 %	0 %
$\alpha = 0,\ \beta = 1,\ \gamma = 0$	μ	42.38 %	2.18 %
	σ	11.54 %	4.187 %
$\alpha = 0,\ \beta = 0,\ \gamma = 1$	μ	2.61 %	0 %
	σ	1.62 %	0 %

relationships are set between PD and GP actions of the individual orange and their neighboring bricks.

Figure 6.18 shows the trajectories of the two UAVs during the execution of the described task. The figure is plotted with respect to time, with the specific actions of the robots (takeoff, going to position, picking, and placing the brick) clearly marked. Synchronization of the UAVs is achieved by modeling their tasks as complex redundant and interconnected action nodes with *precedence* relationships. During execution, the involved robots form a direct communication link to exchange their state and further synchronize all steps of the currently executed action.

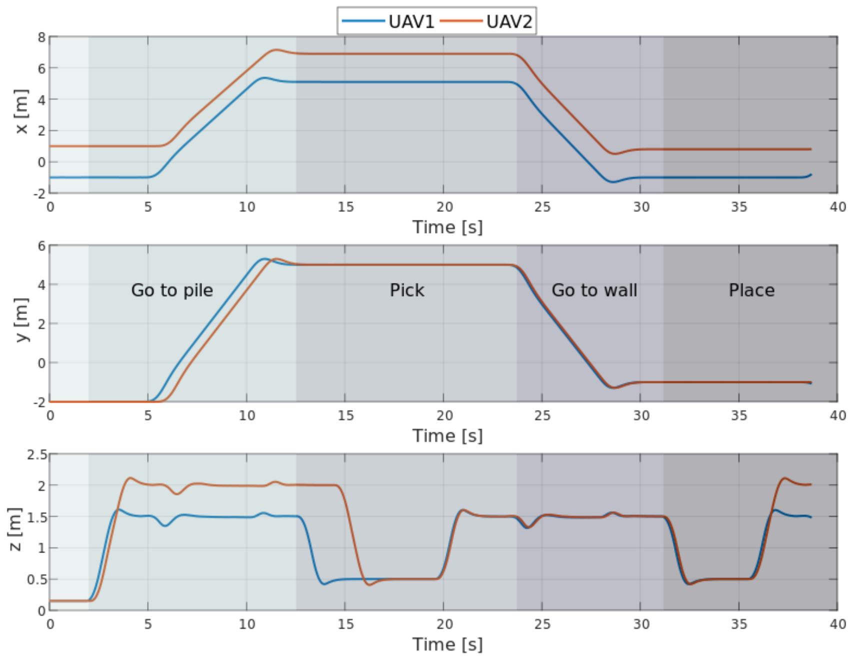


Figure 6.18: Trajectories of UAVs during collaborative transportation of the orange brick.

The video of the simulations can be found at [107]. We also applied the method to the mission of building a wall with a larger number of bricks to illustrate the scalability of the proposed solution. The figure of the resulting schedule is too large to be conveniently included in the thesis itself, but it is available at [108].

In this work, we further extended the capabilities of the proposed coordination framework by introducing more complicated task relationships and the concept of resources. The results show that our method can successfully generate schedules for cooperative missions with highly mutually constrained tasks under different setups of a multi-criteria objective. However, the task allocation and scheduling procedures are still uncoupled, which we identify as the next point for improvement. Therefore, in our following work, we focus on modeling these problems and propose a fast and efficient solution as described in Sections 4.3 and 5.1.

6.4 Distributed Mission Planning for Problems with Cross-Schedule Dependencies

In this section, we analyze the performance of the algorithm proposed in Section 5.1 for task planning problems defined in terms of VRP, as described in Section 4.3 (Contribution 2). To empirically validate the proposed algorithm, we first evaluate our distributed solution on the benchmark Multi-Depot Vehicle Routing Problem (MDVRP) dataset[109] and its solutions available in the VRP-related literature. However, these benchmark problems do not cover all the complexities of the considered *task planning problems*, such as precedence constraints. Therefore, we generate a generic set of multi-robot task planning problems and compare our solution on them against the following techniques:

1. Gurobi Optimizer [44]. Gurobi is a centralized solution that uses exact mathematical methods to solve MILP problems. Gurobi provides an optimality gap for each solution and therefore can be used as a measure of the optimality of the proposed solution.
2. State-of-the-art distributed auction-based approach with a single central agent acting as an auctioneer [55]. We slightly adapt the method to suit our problem class.

Gurobi can only generate solutions for the simpler problems in our dataset because its computational complexity increases rapidly with the dimensionality of the problem (larger number of tasks). From a very limited pool of distributed approaches in the literature, we selected the auction-based method [55] as a direct counterpart to our method. We have chosen the given approach because it is the most recent solution from the literature that covers all the constraints we are interested in and that can be reproduced and implemented based on the information provided in their paper. These two methods serve as benchmarks against which we evaluate the performance of our proposed approach.

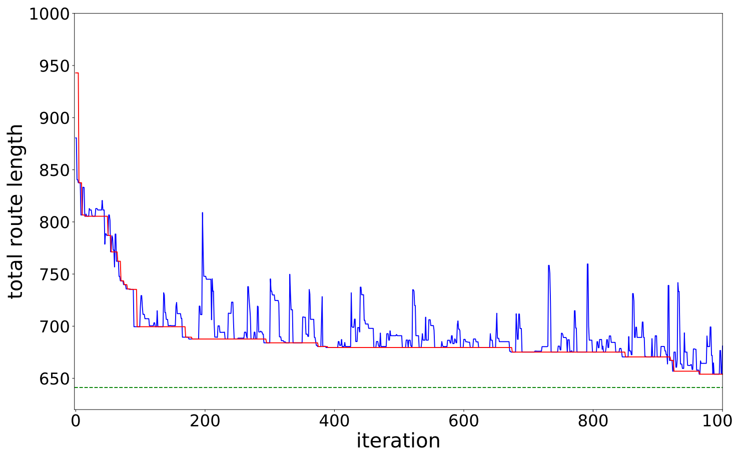
In this section, we discuss the performance results of the proposed algorithm on different problem instances compared to the state of the art and optimal solutions. The main properties we are interested in are optimality, computational speed and scalability. First, we analyze the proposed population-based CBM (Algorithm 2) compared to the original single-solution version of the CBM algorithm. Next, we study the performance of the proposed CBM-pop on an established set of benchmark examples for MDVRP. We then test the algorithm on a large scale on randomly generated task planning problems with precedence constraints and transitional dependencies (including the cost and time limitations of switching between any two actions). We compare the results with optimal solutions obtained using the Gurobi solver and a state-of-the-art auction-based planner. Finally, we demonstrate the performance of the algorithm in a realistic planning scenario for routing mobile robots in a greenhouse setting.

The proposed solution is implemented in ROS environment to enable vehicle-in-the-loop development of planning algorithms. This facilitates the rapid transition from the simulated environment to a real-world experiment. ROS also provides the underlying communication infrastructure through its message and service protocols. All our simulations were run on Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz x 8, 32 GB RAM running Ubuntu 18.04 LTS operating system.

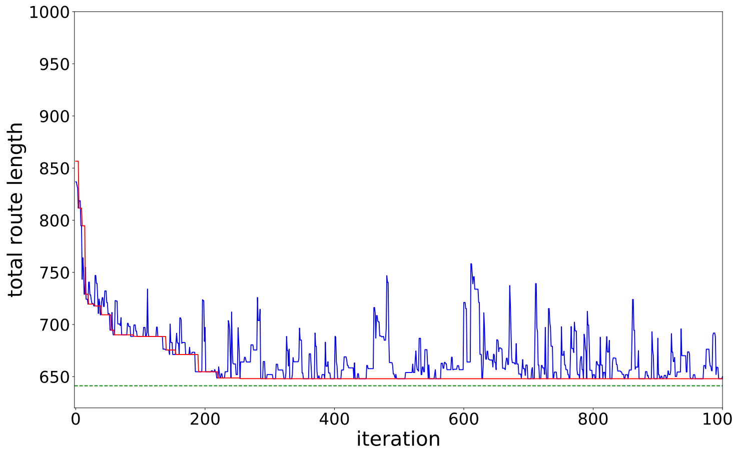
One of the important factors affecting the performance of the algorithm is the selection and tuning of the parameters. In this paper, we define the parameters empirically, following some common practices in evolutionary computing. First, the population size defines the diversity of the algorithm's solution pool. While a large diversity is good, larger populations lead to slower convergence of the algorithm. This parameter should be set considering the known properties of the problem to be solved. In our simulations, we set the *pop_size* to 50. Next, the reinforcement learning factors $\boldsymbol{\eta}$ define the learning rate applied to the weight matrix after finding the best agent (coalition) solution. We set this parameter to $[0, 5, 1]$, which determines a 50% larger weight increase for the case where the best coalition solution has improved compared to the only locally best solution. The mimetism rate ρ, which determines the rate of an agent's mimetic behavior, is set to 0.3, where an agent mimics another agent's weight matrix by 30% and retains 70% of its own learned behavior. Finally, the termination parameters *n_cycles* and ϵ are set specifically for each of the application scenarios. A typical value for *n_cycles* would be no less than 10000 to obtain high quality solutions, while ϵ represents the smallest allowable improvement in the solution and we used the value of 0.01 to run the algorithm as long as there is progress.

6.4.1 Performance of the population-based CBM

First, we evaluate the performance of the proposed CBM-pop compared to the single-solution variant of CBM [91]. In Figure 6.19 is the visual representation of the current (blue) and best solution (red) trajectories for the two variants of the CBM algorithm on a benchmark problem for MDVRP. The two plots illustrate the contrast of the two approaches in the early search of the solution space. The CBM-pop manages to quickly jump through various solution configurations and explore different local optima, leading to faster convergence towards the optimal region. In Figure 6.19b, we can observe a rapid convergence (within the first 200 iterations of the algorithm) of the found solutions towards the dashed green line, which is the best known solution for the given problem. In contrast, the solution trajectories in Figure 6.19a show a noticeably slower progress towards better solutions. This difference is the direct result of the genetic diversity of the obtained solution pool of CBM-pop and is consistent with the intended algorithm design.



(a) original CBM [91]



(b) population-based CBM

Figure 6.19: Illustration of typical trajectories of the current (blue) and the best found solution (red) for the two variants of the CBM algorithm. We analyze the convergence speed of single-solution CBM and the proposed CBM-pop. The best known solution is shown in the graph with a dashed green line.

6.4.2 Performance on benchmark examples of MDVRP

Given the compelling similarity between the problems of capacitated MDVRP and task planning, we first analyze the performance of the developed algorithm using well-studied benchmark examples of MDVRP. The benchmark problems we regard [109, 110] require the assignment of customers to depots and the routing of individual vehicles. The problem solution must serve all customers while minimizing the distance traveled. There is also a maximum route capacity assigned to each vehicle.

We compare the performance of the CBM-pop algorithm with two recent operations research papers that address the same problem on the Cordeau benchmark dataset. The first algorithm selected is the Tabu Search heuristic (TSH) algorithm presented in [90]. The TSH algorithm was chosen for comparison as a centralized state-of-the-art heuristic algorithm from the field of operations research. The other selected algorithm is a distributed

Table 6.7: Performance comparison between CBM-pop and two state-of-the-art optimization algorithms TSH [90] and CoEV [89] on Cordeau benchmark examples [109] for MDVRP. In the table, m and n represent the number of vehicles and customers, respectively, for each test example. The best solutions found for all algorithms are given, as well as the optimality gap (%) with respect to the best known solutions (BKS) for the benchmark defined in [109]. The average optimality gap is given for CBM-pop based on 50 runs of the algorithm. The total CPU times are given for algorithms in the defined simulation setup. We estimate the computation time per vehicle $\tau(s)$ when CBM-pop is run fully distributively on each of the vehicles in the system. The best solutions are highlighted in bold.

				TSH[90]			CoEV[89]			CBM-pop				
test	m	n	BKS	best	gap(%)	CPU(s)	best	gap(%)	CPU(s)	best	gap(%)	avg. gap(%)	CPU(s)	$\tau(s)$
p01	16	50	**576.87**	**576.87**	0	9.4	**576.87**	0	1.0	**576.87**	0	2	18.1	4.5
p02	8	50	**473.53**	**473.53**	0	20.9	473.87	0	0.5	**473.53**	0	1	12.3	6.2
p03	15	75	**641.19**	**641.19**	0	141.9	**641.19**	0	2.5	**641.19**	0	1	26.1	7.0
p05	10	100	**750.03**	758.87	1.16	159.1	750.11	0	26.6	752.05	0.27	2	43.1	17.2
p06	18	100	**876.50**	881.76	0.60	194.8	**876.50**	0	77.3	893.59	1.91	3	44.0	9.8
p09	36	249	3900.22	3971.59	1.80	606.0	**3895.70**	**-0.12**	513.30	4049.60	3.69	9	97.7	10.9
p10	32	249	**3663.02**	3779.10	3.07	703.7	3666.35	0	719.90	3792.10	3.40	8	117.3	14.7
p11	30	249	**3554.18**	3652.01	2.68	660.3	3569.68	0.43	396.20	3713.39	4.29	8	122.1	16.3
p12	10	80	**1318.95**	**1318.95**	0	13.8	**1318.95**	0	0.9	**1318.95**	0	1	24.1	9.7
p13	10	80	**1318.95**	**1318.95**	0	6.7	**1318.95**	0	0.0	**1318.95**	0	0	24.6	9.9
p15	20	160	**2505.42**	2552.79	1.86	255.4	**2505.42**	0	432.0	2565.40	2.34	4	57.6	11.5
p18	30	240	**3702.85**	3802.29	2.62	302.9	3771.35	1.82	429.1	3814.23	2.92	6	159.6	21.3
p21	45	360	**5474.84**	5617.53	2.54	1703.0	5608.26	2.38	554.9	5731.88	4.48	7	267.3	23.8
pr01	4	48	**861.32**	**861.32**	0	2.1	**861.32**	0	0.0	**861.32**	0	0	4.8	4.8
pr09	18	216	2153.10	2177.20	1.11	613.5	**2150.52**	**-0.12**	1107.15	2187.84	1.59	5	75.5	16.8

cooperative coevolutionary algorithm (CoEV) [89]. This algorithm runs in a distributed manner, where for each depot a partial solution is constructed and optimized. The TSH algorithm was run on Intel Xeon (R) E5-2690 v4 x 14 processor (2.60 GHz, 32 GB RAM), while the CoEV was run on a computer with two 2.50 GHz Intel Xeon (E5-2640) processors with 12 cores per CPU and 96 GB RAM, both much more powerful computers than the one we used in our results. Since the algorithms run on different hardware platforms, the computation times should only be used for illustration. Nevertheless, we want to show the suitability of the proposed algorithm as an MDVRP solution compared to the state of the art, both in terms of optimality and scalability (runtime).

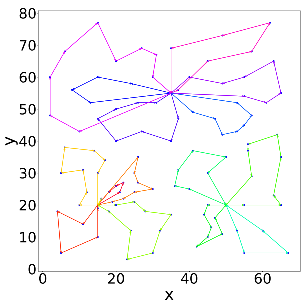
CBM-pop is a distributed algorithm that should typically run on each robot during the mission runtime. For comparison with the TSH and CoEV algorithms, the CPU times in Table 6.7 are the times required to compute the contribution of all robots in CBM-pop on only one computer. Therefore, CPU times reflect the total computational effort for all algorithms. On the other hand, $\tau(s)$ is the runtime per vehicle when CBM-pop is executed distributively in each vehicle. Thus, the runtimes of CBM-pop are given by τ, while for the other two algorithms they are given by the CPU times. CoEV simulations are originally run fully distributively, and the given CPU times reflect the actual performance of this algorithm.

The full simulation results are shown in Table 6.7. The benchmark examples range from 2 to 6 for the number of depots, with 2 to 14 vehicles per depot, and customer numbers from 48 to 360. In Table 6.7, the performance of algorithms is directly related to the best known solutions (BKS) for each test example in [109], and the optimality gap for best simulation runs is also given.

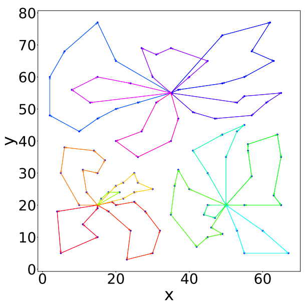
We can conclude that the algorithms CBM-pop and TSH give very similar results in terms of optimality, both well within 5% of the best solutions found for the problem. However, since CBM-pop is distributed, the total CPU effort is shared between all involved robots, resulting in significantly lower runtimes (given by $\tau(s)$). On the other hand, CoEV shows the best performance in terms of optimality of solutions. Although this algorithm is distributed, it shows significantly higher runtimes for examples with more vehicles and customers. The possible reason is that the subproblems become more complex, so the distribution of the problem per depot shows less influence on the performance of the algorithm. In summary, our proposed algorithm CBM-pop produces solutions that are on par in optimality with state-of-the-art operations research approaches to this problem. Due to the distributed nature of CBM-pop, it generates solutions in less runtime, which makes it suitable for online applications for mission planning problems. Another major advantage of CBM-pop is its robustness to agent breakdowns since the computations are performed on multiple nodes.

These tests indicate that CBM-pop provides a very good solution to the given problem and does so in a short computation time. It is important to note that these benchmarks were developed as a very complex test for optimization algorithms and that the time taken to compute the solutions is usually not crucial. For us, the timeliness of the solutions is essential since we are dealing with implementation on a robotic system. With this test, we have shown that our distributed algorithm can handle cases with higher complexity than we expect in robotic systems in a relatively short time.

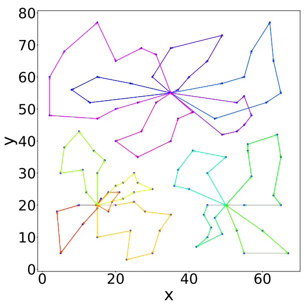
To better illustrate the test set and the results obtained, we graphically represent in Figure 6.20 the best solutions found for three test examples from the benchmark set. The full simulation results are available on the results webpage [111].



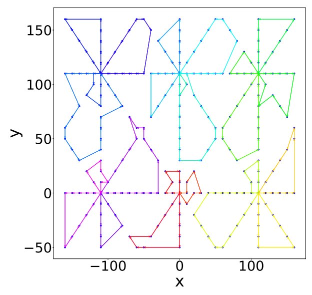
(a) p06 GA best of 50 runs



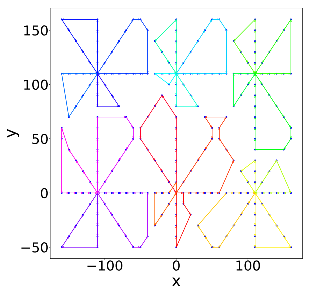
(b) p06 CBM-pop best of 50 runs



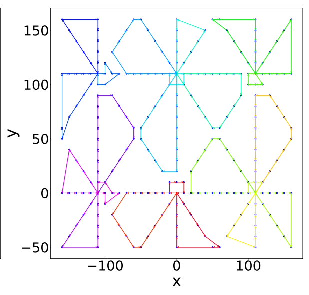
(c) p06 best known solution



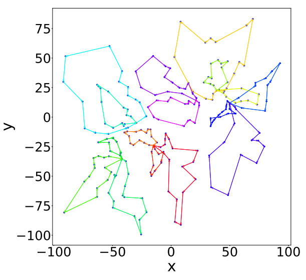
(d) p18 GA best of 50 runs



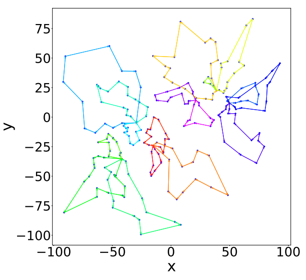
(e) p18 CBM-pop best of 50 runs



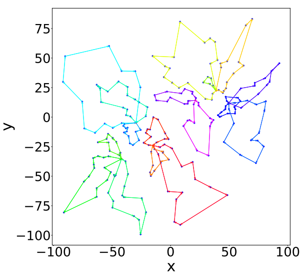
(f) p18 best known solution



(g) pr09 GA best of 50 runs



(h) pr09 CBM-pop best of 50 runs

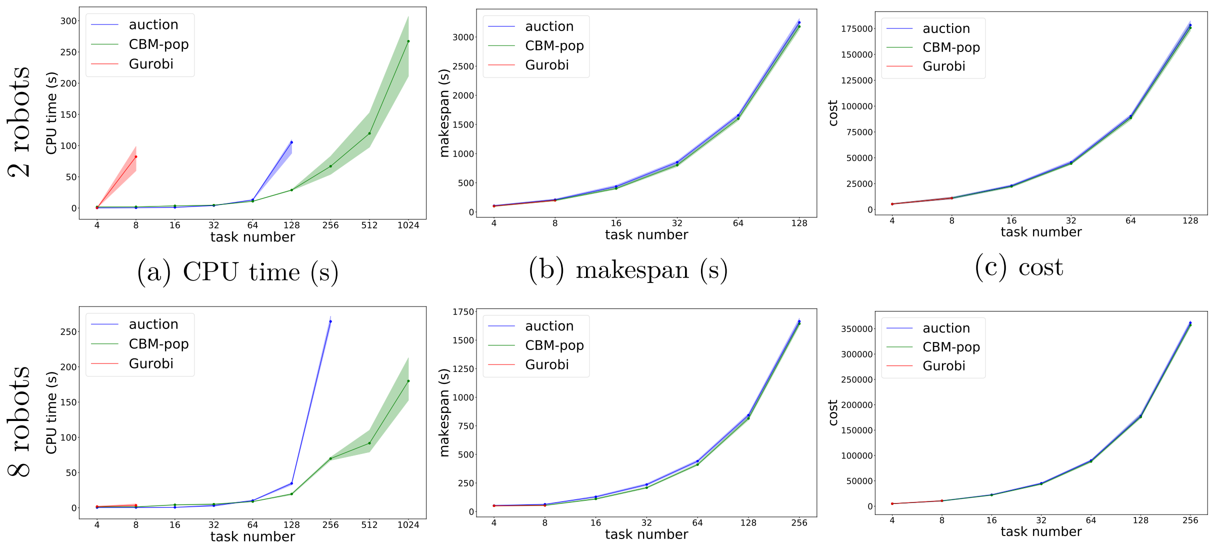


(i) pr09 best known solution

Figure 6.20: Example solutions of Cordeau benchmark tests [109] for genetic algorithm, population-based CBM, and best known solution from the literature.

6.4.3 Comparative analysis on tasks with cross-schedule precedence constraints

Despite the considerable complexity of the previous tests, they do not include some important elements of the problem that our approach solves, namely cross-schedule dependencies. Therefore, we have developed a separate set of benchmark examples to evaluate problems with precedence constraints. The problems are solved for a team of 2 and 8 robots. The problem instances include 4, 8, 16, 32, 64, 128, 256, 512, and 1024 tasks, and we generated 50 randomized examples for each setting. In each example, 20% of the tasks are precedence constrained. Task durations and costs are generated using assumed robot characteristics (speed, energy requirements). Setup times and costs are calculated using the above mentioned robot characteristics and the Euclidean distance between tasks associated with random positions in 3D space. The full benchmark set and results can be found at the benchmark website [112].



(a) CPU time (s) (b) makespan (s) (c) cost

(d) CPU time (s) (e) makespan (s) (f) cost

Figure 6.21: Comparison of the performance of the proposed distributed metaheuristic algorithm CBM-pop (green), the Gurobi optimal solver (red), and the state-of-the-art auction-based distributed algorithm [55] (blue). The top row shows the results for 2 robots, and the performance for 8 robots is shown in the bottom row. The plot displays a graph for the mean of each of the observed values and the distribution highlighted by a transparent ray around the graph line. A time limit of $20min$ was introduced in the simulations, and only the results obtained in this runtime are presented.

In analyzing the results, we compare the proposed distributed metaheuristic CBM-pop with an exact MILP-based solution provided by the Gurobi solver [44] and a state-of-the-art distributed auction-based algorithm presented in [55]. The performance results of the algorithm for the defined benchmark set are shown in Figure 6.21.

Two rows of the figure represent the performance of the algorithms for teams of 2 and 8 robots, respectively. In the simulations, we introduced a computation time limit of 20 minutes. The optimal solver is able to obtain solutions for up to 8 tasks, the auction-based algorithm, for up to 128 tasks for the case of 2 robots, and 256 tasks for 8 robots, while the proposed algorithm can handle all examples in the benchmark. For the case of 2 robots and 256 tasks, and 8 robots and 512 tasks, the auction algorithm takes about 25 minutes to produce solutions.

The first property to be observed is the algorithm runtime and scalability of the above approaches. From the first column of the grid in Figure 6.21, it is clear how fast the combinatorial explosion manifests in the auction-based algorithm. It is even clearer for the optimal solver. The Gurobi solver succeeds on problems with up to 8 tasks for both sets of benchmarks. The auction method is able to solve problems with at most 256 tasks in the given time. An exponential increase in computation time can be observed. On the other hand, CBM-pop copes very well with an increase in the number of robots and tasks. Also, a larger scatter in the CPU time in CBM-pop is observed for larger task examples. This is due to the stochastic nature of the protocol and the quality-based stopping criterion, which terminates the computation if no improvement in the solution has been achieved for a certain number of steps.

Table 6.8: Summary of average improvements of the proposed algorithm (CBM-pop) compared to the state-of-the-art auction-based algorithm [55] on a basis of 50 randomized examples for each problem setting.

	task number	average improvement (%)		
		makespan	cost	CPU time
2 robots	4	4.36	0	-
	8	5.25	0.51	-
	16	7.45	3.12	-
	32	5.03	2.36	-
	64	3.39	1.86	15.89
	128	2.39	1.57	71.60
8 robots	4	4.57	0	-
	8	14.39	0	-
	16	15.04	2.57	-
	32	11.48	2.70	-
	64	7.00	1.93	12.76
	128	3.43	1.27	42.35
	256	1.32	1.1	73.11

In Table 6.8, we can observe the average improvement of the CPU time of the proposed algorithm compared to the auction-based method. We provide information for more extensive problems where the qualities of the proposed method are highlighted. For simpler examples, the auction-based algorithm renders solutions faster (about 2-3 times). CBM-pop computes solutions in about 1.5s for the simplest cases (5s for more complex problems), and the auction manages to solve the problem in approximately 0.5s and 2s, respectively.

We also investigate the performance of the algorithms in terms of optimality. As explained above, we model the optimization function in terms of Pareto optimality (as defined in Eq. 5.5) for two criteria, namely the makespan of the schedule and the total cost (Eq. 4.14.2). For the limited example set with computed optimal solutions, both the auction and our method follow the optimal solutions very closely (well within 0.5% of the optimum). In Figure 6.21, the last two columns represent the duration and cost of the solution found. In all examples presented here, the CBM-pop algorithm outperforms the auction for both given criteria. For the case of 2 robots, the improvements range in makespan from 2% to 7% and in cost up to 3.12%. For 8 robots, the improvements range in makespan from 2-16% and in cost up to 2.6%. All results are summarized in Table 6.8.

By running these simulations, we have shown that our approach can keep up with the current state of the art in task planning in terms of optimality, while generating solutions in significantly less time, which is essential for all real-world applications.

6.4.4 Application to the use-case in agricultural environment

The use of robots in agriculture is not a new concept, but rather a rapidly growing industry that focuses primarily on large machines used for specific crops and applications. However, the main objective of the SpECULARIA project is to develop a heterogeneous robotic system consisting of three types of agents: an Unmanned Aerial Vehicle (UAV), an Unmanned Ground Vehicle (UGV), and a compliant manipulator with several degrees of freedom.

The primary role of the UAV in the system is to monitor the system, where it can issue possible maintenance actions. The UGV is equipped with a mechanism that allows it to transport containers of growth units, which are the smallest organizational unit within the farm and consist of a single plant or a variety of plants. The task of the compliant manipulator is to perform delicate manipulations on plants, such as handling flowers and fruits and pruning plants. Each robot has several specific capabilities, but when used together, they can be used in different ways to accomplish multiple objectives.

The mission given to our robotic team is to perform daily maintenance tasks in a robotized greenhouse. The team consists of three unmanned ground vehicles (UGVs) equipped with a mechanism to transport plant containers to and from the workspace and a single robotic manipulator that performs actions on plants. Each UGV can pick up and transport one plant at a time and place them on specific tray holders in the greenhouse. We assume that missions to the system are issued in time frames greater than the time required to complete a single mission, i.e., the team has enough time to complete one mission before the next one arrives.

The layout of a greenhouse in a mission we analyze in this section is shown in Figure 6.22. The greenhouse consists of two tables with plants placed along the walls of the greenhouse. Each table is organized into five rows and two columns. The tables are enumerated, as are the specific positions within the table. The convention for addressing the tray holders within the table is *(row, column)* and the indices begin with 0. The full

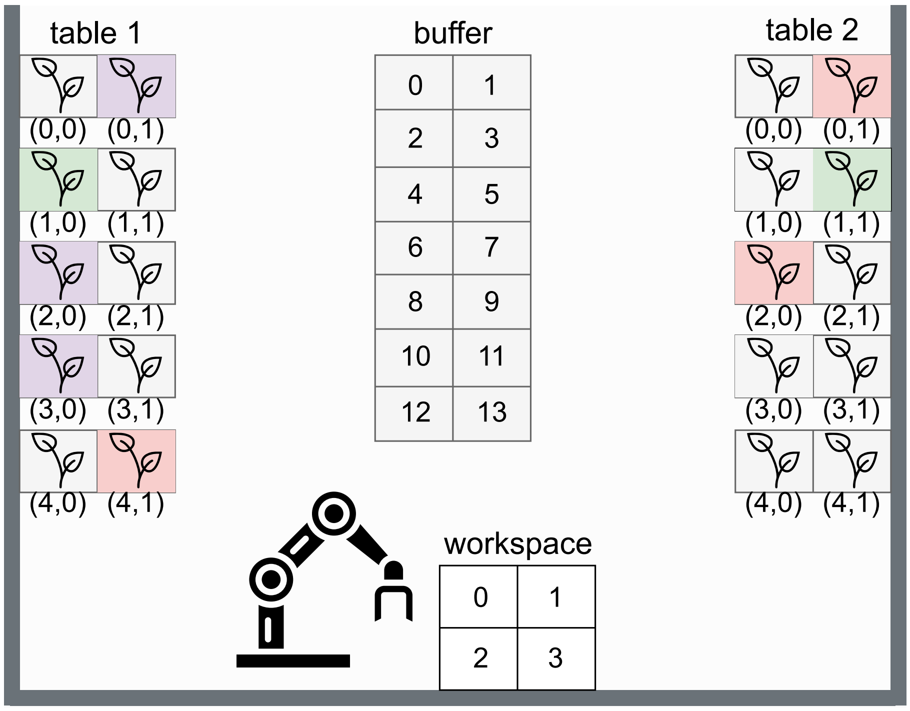


Figure 6.22: SpECULARIA use case greenhouse layout.

address of each plant is defined by the triplet *(table, row, column)*. Plants within the table can only be accessed by row, starting with the positions at the aisle. Thus, plants located by the wall of the structure can only be accessed by removing previous plants in the same row of the table. For example, in Figure 6.22, the plant at position $(1, 0)$ in table 1 can only be accessed after removing the plant at position $(1, 1)$ from the table. A similar precedence relationship applies to table 2, except that plants are accessed from left to right.

In addition to the two tables, there is a buffer table structure in the middle of the greenhouse. The structure of a buffer is very similar to the structure of the tables, but the plants can be reached from either side, so there is no precedence relationship between the plant access tasks. The buffer is used to store plants that need to be put aside before the required plants are transported to the processing station (work station of the robot manipulator). Plants that are finished with the maintenance task are also put back into the buffer.

Finally, at the bottom of the greenhouse structure is a workspace table with four plant tray holders. The idea of the four positions is to allow for batch processing of the plants, which is especially advantageous for simpler tasks such as watering or spraying the plants.

Inputs to the planning procedure include the greenhouse layout, the groups of plants to be tended that day, and the procedures to be performed. In the given example from Figure 6.22, the specific groups are $A = \{(1, 0, 1), (1, 2, 0), (1, 3, 0)\}$, marked in purple in the figure, $B = \{(1, 1, 0), (2, 1, 1)\}$ marked in green, and $C = \{(1, 4, 1), (2, 0, 1), (2, 2, 0)\}$

marked in red. This means that to execute operation A, all three defined plants must be present in the workspace table. The same is true for the other two tasks.

In problem modeling, we distinguish between two actions that the UGV can perform on plants, namely transporting plants to the buffer and moving plants to the workspace. Based on these two actions and the defined inputs, we generate a set of actions to be planned for. For plants that are not scheduled for care on a given day and that interfere with the plants to be processed, the action of moving them to the buffer is generated. For example, we can identify the task *to_buffer(1,1,1)*. For plants that are scheduled for care, we define two precedence-constrained actions, moving them to the workspace and placing them in the buffer when processing is complete. For example, we define the tasks *to_workspace(1,1,0)* and *to_buffer(1,1,0)*, with a precedence constraint in between. Additionally, there is a precedence for the tasks of accessing two adjacent plants, in this example the tasks *to_buffer(1,1,1)* and *to_workspace(1,1,0)*.

On the side of the manipulator that tends the plants, we define three different actions. This was necessary to ensure the desired system behavior while keeping the problem within the scope of the defined modeling. For the maintenance task A example, the defined tasks are *A_ready*, *A_perform*, and *A_setup*, all of which take precedence in the defined order. Task *A_ready* signals that the workspace is empty and ready to receive the next batch of plants. This task precedes all tasks *to_workspace* for the given procedure. After all plants are placed on the workspace, the task to perform the procedure (*A_perform*) begins. This relationship is also modeled by precedence constraints. Next, after the procedure is completed, the tasks of transporting the plants from the workspace to the buffer are activated. After all plants are removed from the workspace, the task *A_setup* is executed. This task represents the tool change of a robot arm. When it is finished, new plants can be brought into the work area and the whole process starts again.

We have run several simulations for the specified use case. As for the benchmark problem set, we compared the performance of the proposed algorithm with the auction-based state-of-the-art solution for task planning problems [55]. We ran 10 simulations for each algorithm, and the results are consonant with the conclusions drawn earlier. In Figure 6.23 we can observe CPU time, makespan, and cost distribution for the obtained solutions. For this example, the average reduction in CPU time of the proposed algorithm compared to the auction protocol is 13.23%. Regarding makespan and cost, a slight average improvement of 0.59% and 0.76%, respectively, is observed.

For this relatively small problem, the qualities of the obtained solutions are very similar for both algorithms. However, we have previously shown that our algorithm adapts better to the problem complexity. Another important advantage is that our solution produces results in a distributed manner. The auction algorithm, on the other hand, assumes a

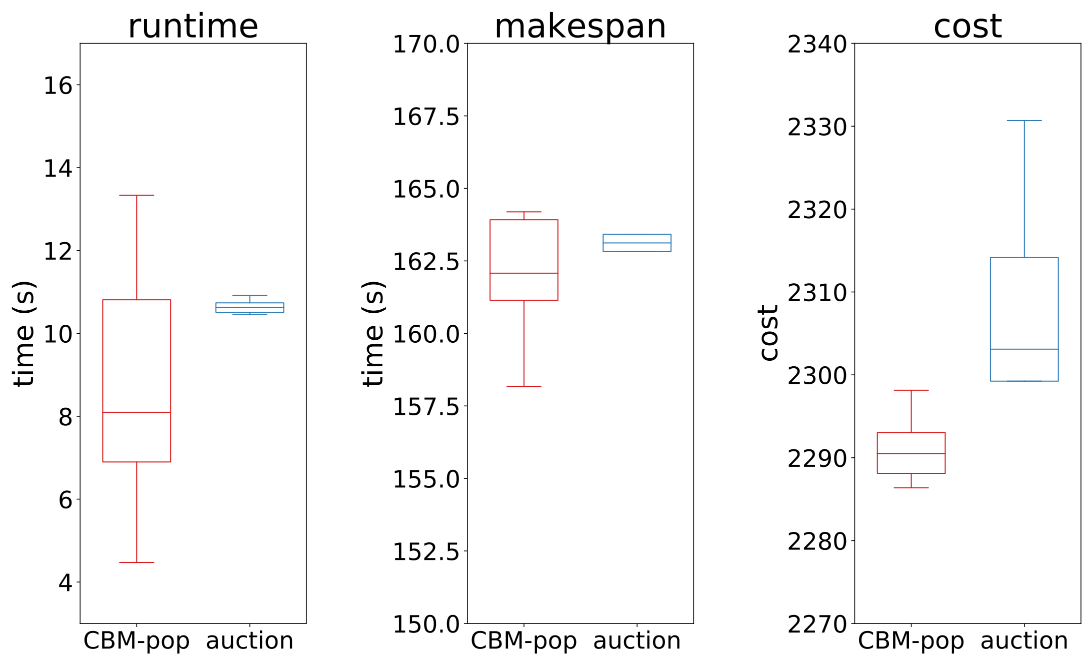


Figure 6.23: Comparison of the performance of the proposed CBM-pop (red) with the auction-based algorithm (blue). We observe the algorithm CPU time, solution makespan, and total cost for 10 runs of each algorithm.

central auctioneer agent that processes bids from other agents and makes task assignments. In distributed systems, this may be regarded as a tactically vulnerable point.

The final generated schedule for the SpECULARIA use case is displayed in Appendix for convenience. Tasks are color-coded according to the convention defined in Figure 6.22. Tasks of moving plants that do not require maintenance into the buffer are marked in gray. Precedence relations are indicated by arrows. Setup times are not displayed in this illustration to preserve a more compact representation of the schedule. Based on the generated schedule, we can conclude that the proposed method successfully handles the problem of planning tasks for this type of operations. The animated visualization of the execution of the obtained schedule is available at [111].

In summary, in this work we developed a robust and fast task planning method for heterogeneous multi-robot systems. The planning method addresses problems with cross-schedule dependencies, in particular precedence constraints. We synthesized a general model that relates task planning (allocation and scheduling) to the well-studied VRP. This exposes task planning problems to various optimization techniques available in vehicle routing, which could lead to many compelling solutions for task planning in the future.

In this work, we have found a solution to the problem in a distributed manner by applying a metaheuristic approach based on evolutionary computation with knowledge sharing and mimetism. We have extensively tested the performance of the proposed algorithm. First, we ran simulations on an established set of benchmark examples of capacitated multi-depot VRP, where the algorithm was found to perform near-optimally, with a high computational speed. Next, we established a benchmark dataset repository for planning for tasks of class XD[ST-SR-TA] and tested the proposed algorithm against existing task planning methods. Simulation results show that the approach has better computational speed and scalability without loss of optimality compared to state-of-the-art distributed methods. We have also provided a novel application of the planning procedure to a real-world use case of a greenhouse maintained by a multi-robot system.

Consequently, we focus our efforts on the next class of problems, complex dependencies (CD class). Although some of the solutions shown so far address problems of this class, they do so in a sequential manner, where task decomposition selection, task allocation, and scheduling are performed separately. In the following section, we present the results of a method that couples the three aspects of mission planning into a single two-step optimization procedure, as defined in Section 5.2.

6.5 Distributed Mission Planning of Complex Tasks

The missions we model in this work fall into the class of problems CD[ST-MR-TA] [38]. These missions involve tasks that may require execution by more than one robot (MR, multi-robot tasks), and robots may only perform one task at a time (ST, single-task robots). The task allocation and scheduling procedure considers both current and future assignments (TA, time-extended assignment). In terms of complexity, these tasks include complex task dependencies (CD), where each task can be achieved in multiple ways. The class CD also entails cross-schedule dependencies (XD), where various constraints relate tasks from plans of different robots.

To represent multi-robot missions, we use a hierarchical task model inspired by TÆMS. In our solution, missions represented as large task hierarchies are subjected to a two-stage hierarchical optimization procedure. In the first step, we perform a fast and efficient heuristic search of the mission tree that finds several promising alternative ways to execute the mission (task decomposition selection procedure). Then, a task allocation and scheduling procedure is applied to several best-ranked alternatives to generate schedules for the given problem. Based on the given criteria, the best overall solution is output as the final schedule that best satisfies the mission objective. The full procedure is described in Section 5.2.

The method is evaluated in a simulation setup of an automated greenhouse use case, where we demonstrate the method's ability to adapt the planning strategy depending on the available robots and the given optimization criteria.

6.5.1 Simulation setup

To evaluate the performance of the proposed method, we developed a practical use case with problems of class CD. The application is based on an automated greenhouse and the scheduling of its daily maintenance tasks. The greenhouse structure is organized as a set of tables, comprising several plant containers representing growth units. In this example, each container holds a single plant that can be conveyed through the greenhouse using a UGV with a special mechanism for transporting plants.

The UGV works in symbiosis with a stationary manipulator located at a workspace with four empty container holders. The manipulator can perform various operations on the plants once they are brought into the workspace. The design of a workspace with multiple slots allows batch operations on plants, speeding up some of the procedures. Naturally, not all tasks support batch processing to the same degree.

In addition to the stationary manipulator, we also envisioned a mobile manipulator consisting of a larger UGV with a robotic arm on board. This robot is capable of driving around the greenhouse and tending the plants directly. Note that each of the operations in this case must be performed from both sides of the table to take into account the entire plant.

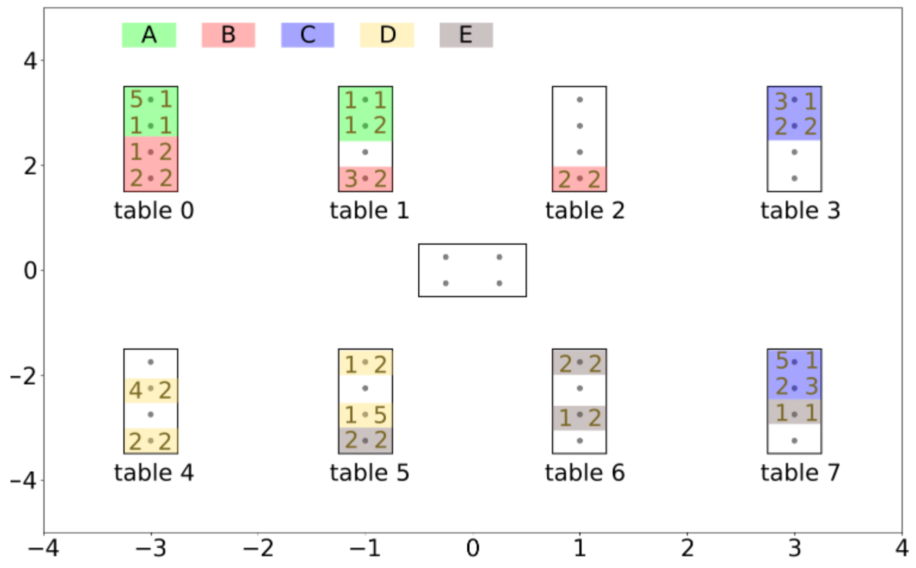


Figure 6.24: Greenhouse use case plant setup. In the illustration, the different batches of plants are marked in different colors, as indicated in the legend. The dimensions of the greenhouse (in meters) are shown on the x and y axes.

The structure of the greenhouse we used for the simulations is shown in Figure 6.24. The structure consists of eight tables, each with 4 plants. For the purpose of batch processing, the plants were clustered a-priori into five groups $\{A, B, C, D, E\}$ as shown in the figure. In this arrangement, there are a total of 20 plants that need to be tended to. The numbers given in the table structures for each plant denote the number of unit operations that need to be performed on each side of the plant. They are used to estimate the total procedure duration. In our example, we specify the unit operation duration as $10s$. Therefore, using the example of the top plant in *table* 0, the total processing time on the left side is $50s$ and on the right side is $10s$.

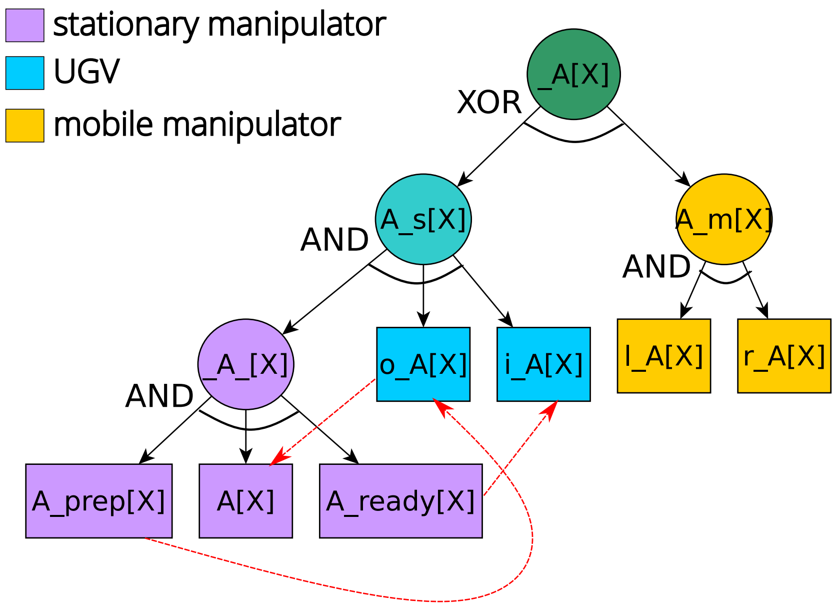


Figure 6.25: Hierarchical task structure for the task of processing a single plant in the system. A is the plant type symbol and X stands for a unique plant identifier, which is composed as *"table.row.column"* of the plant in the greenhouse structure.

The described mission is modeled as a hierarchy of tasks, as described in Section 3.1. A task tree for a single plant operation is shown in Figure 6.25. At the root of the structure is a task representing the desired operation, and it is divided into two subtasks denoting two alternative ways of processing the plant. The left variant defines a stationary case where a UGV and a static manipulator perform the task together. The UGV has to deliver the plant to the workspace (o_A) and return it when the task is finished (i_A). For the stationary manipulator, we defined three tasks, A_prep, A, and A_ready. These tasks are executed sequentially, and the *prep* and *ready* tasks are used to synchronize the operations of the batch procedure with other plants of the group. On the other hand, there is an option where a mobile manipulator tends to the plant, and it includes tasks of the left (l_A) and right (r_A) processing. The full mission structure includes 20 models of this structure, one for each plant, and they are associated with operator *AND*.

6.5.2 Simulation results

We ran several simulations for the setups defined in Table 6.9. The first two setups compare using a single stationary manipulator and two UGVs, versus using two mobile manipulators (problem of class XD). For the next three setups, we used one of each of

Table 6.9: Simulation setups for the use case.

setup	problem class	mobile manipulator	stationary manipulator	UGV	makespan-cost importance
1	XD	-	1	2	-
2	XD	2	-	-	-
3	CD	1	1	1	0-100
4	CD	1	1	1	50-50
5	CD	1	1	1	100-0

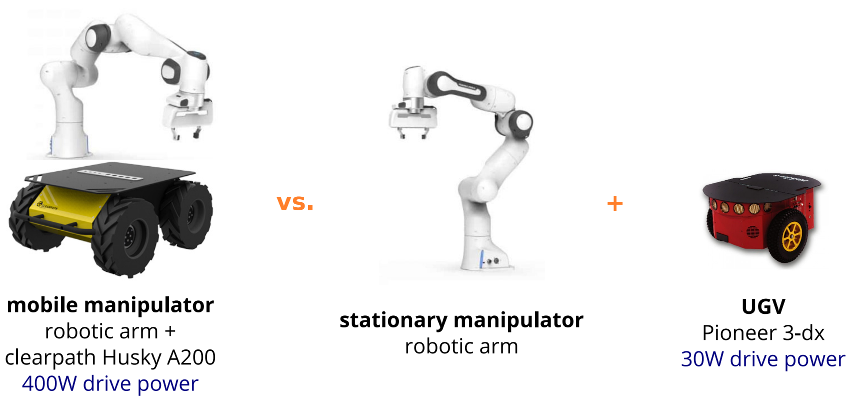


Figure 6.26: Robot team setups in the simulation scenarios. We compare the performance of a mobile manipulator with that of a stationary manipulator supported by UGVs. For complex missions, we allow the use of combinations of all three robots.

the three robot classes but varied the importance of mission makespan and cost in the evaluation procedure. Makespan is defined as the elapsed time between the start and finish of a sequence of tasks in a group of robots.

To estimate the cost of each task, we assume a realistic set of robots, where for a UGV, we suggest a Pioneer 3-DX robot with an estimated drive power of $30W$, while in the case of a UGV carrying a manipulator, we propose a more robust solution of a Clearpath Robotics Husky A200 with a drive power of $400W$. An illustration of the robotic system configurations is given in Figure 6.26. The maximum speeds for both robots are set to $0.5m/s$, taking into account the sensitive payload they have on board. Based on the duration of each task obtained using a path planner on a lower level, we calculate the energy consumed in kJ by multiplying the duration and the power demand. We use a state-of-the-art sampling-based RRT* motion planner. This is a simplified form of a cost function and serves the purpose of testing the planning system. For a more accurate model, the different power requirements of the robots depending on the actual battery charge should be considered.

The results of the simulation runs for each setup are shown in Figure 6.27. There

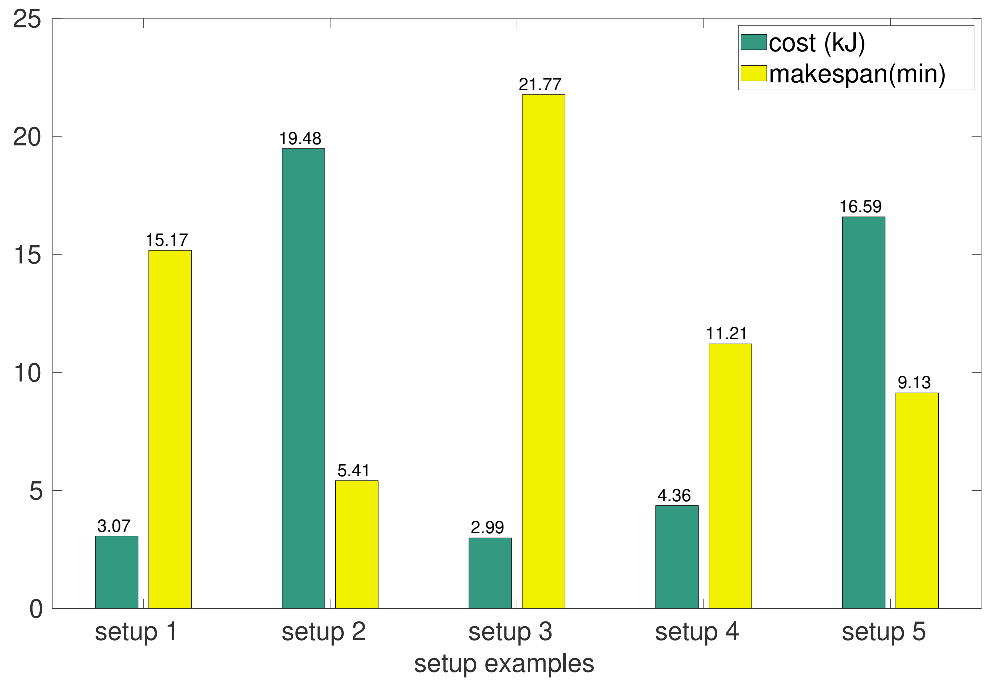


Figure 6.27: Makespan and cost of best found solutions for each setup.

is a clear discrepancy in the ratio of makespan to cost for the first two configurations, where the setup with one stationary arm and two light UGVs consumes much less energy, with the mission time span increasing 2.8 times. On the other hand, it can be seen that the second configuration consumes 6.3 times more energy for execution, although it is faster. To exploit the strengths of both robots, we consider the case of combining these two approaches in the following setups. The proposed mission decomposition selection procedure allows to choose the best way of mission execution considering a certain criterion. Here we analyze three extreme cases: the preference of cost savings in *setup 3*, the equal importance of cost and makespan in *setup 4*, and finally the preference of mission speed without considering cost in *setup 5*.

The results for these setups demonstrate the ability of our proposed complex mission planning system to select appropriate mission decompositions given the robots available in the system and the specified criteria. In a case where the cost is evaluated higher, all tasks are selected to be executed in an energy-efficient manner, which corresponds to the left branch of the mission tree defined in Figure 6.25. For the case where fast execution is required, right branches of the mission tree are selected for all the plants, which are handled by a mobile manipulator. For the middle case, the procedure outputs a solution that balances the tasks between these two options.

Efficiency of the alternative generation algorithm

The next point we consider in the simulation analysis of the proposed method is the computational efficiency of the alternative generation algorithm (Algorithm 3). In combinatorial optimization, this problem is considered NP-hard and no exact solution can be found for it. Our proposed approach uses approximations and heuristics to simplify the problem and solve it in a tractable time.

On an example of an automated greenhouse with 20 plants, we ran 30 simulations for each variant of the algorithm, depending on the degree of approximation in generating partial solutions to the problem. We compare the variant with pruning, which allows at most 10 and 5 alternatives for each task, with a baseline algorithm without pruning procedures.

Table 6.10: Average runtime of alternative generation algorithm for different pruning setups.

setup	average runtime (s)	generated solution no.
no pruning	23.96	1,048,576
prune 10	0.04	1000
prune 5	0.02	125

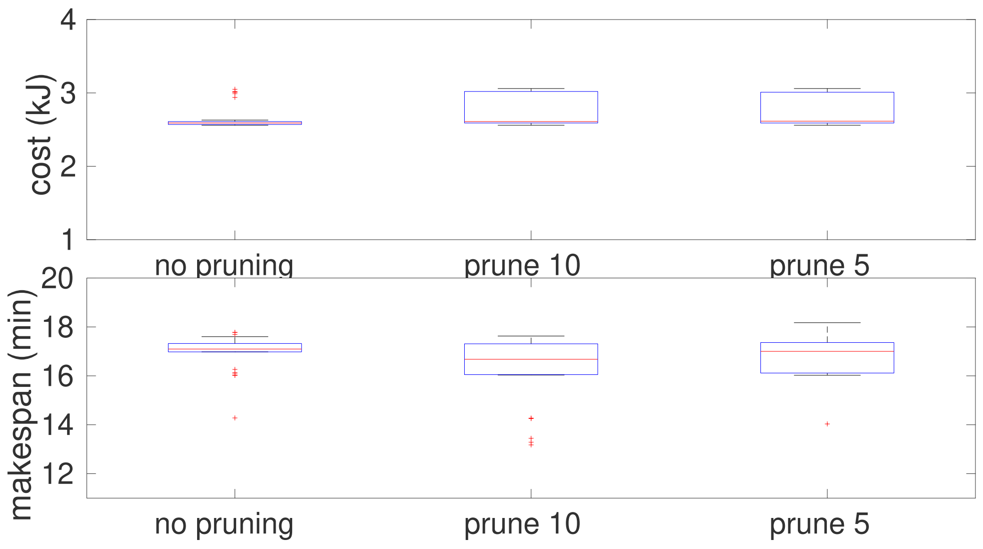


Figure 6.28: Performance of the mission planning algorithm using three different pruning strategies during mission decomposition selection procedure. Score of the alternatives is obtained using $(\alpha, \beta, \gamma) = (0.1, 0.2, 0.7)$.

In the results from Table 6.10, we can observe a significant decrease in the average runtime of the algorithm for this set of simulations. It is important to note that even for this seemingly simple example of 20 tasks with two alternative executions, the total combinatorial number of solutions is 2^{20}. Even in the case of the algorithm without pruning, heuristics and approximations can tame this very difficult problem and solve it in a reasonable amount of time (under $25s$). Both of the pruning variants presented here drastically reduce the computing time (down to $0.04s$ and $0.02s$).

However, this increase in speed needs to be observed through performance perspective as well. In Figure 6.28, we present the makespan and cost of schedules obtained in the second phase of the algorithm for these three setups. The results show no significant decrease in performance when pruning is introduced. For the criteria function which favors the cost component, the solutions found using pruning procedure are all within 13% of the mean value of benchmark solutions.

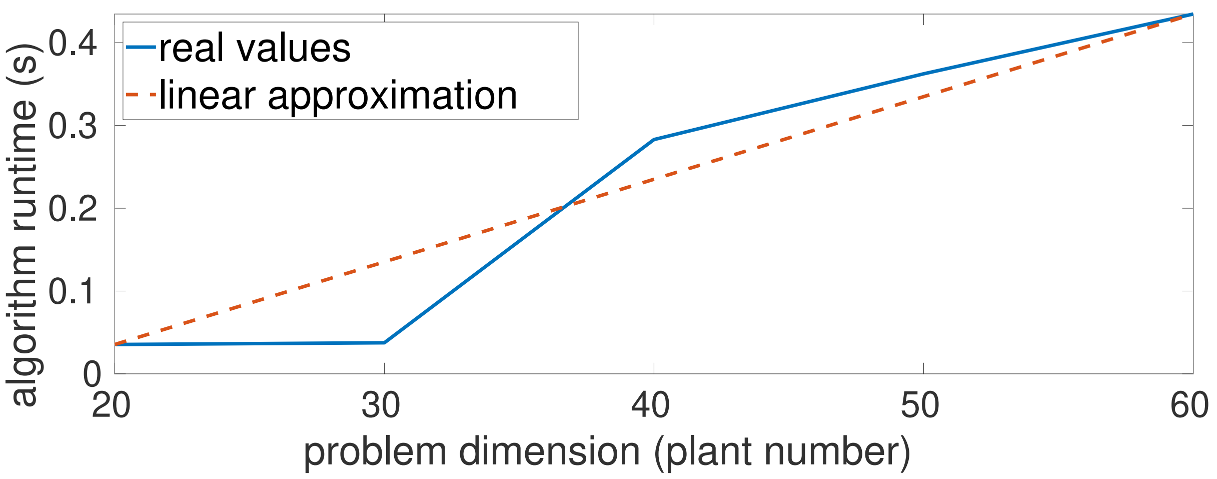


Figure 6.29: Runtime of the algorithm with respect to problem dimension.

Finally, to evaluate the scalability of the proposed heuristic algorithm, we developed larger application examples for greenhouses with 30, 40, 50, and 60 plants. We ran alternative generation procedure (Algorithm 3) with *prune 10* setting for each of the setups. Figure 6.29 shows how the runtime of the algorithm changes for each specific setting. From the obtained data, we can deduct a linear relation between problem dimensionality and algorithm runtime of a very light slope of 0.0998 (depicted in red dashed line in the figure). The nature of this function can, however, dramatically change if the mission structure is not well thought out. In some cases of poorly designed mission trees, this approximation can reach polynomial complexity. It is also important to note that the number of robots does not affect the performance of the alternative generation algorithm.

The main contribution of this work is the proposal of a fast and efficient distributed method for planning complex missions for heterogeneous MRS. The proposed multi-stage optimization approach provides a domain-independent solution to the given problem and can be readily applied to many areas of robotics research that involve cooperative robot teams. The method can adapt the planning strategy and select the appropriate tasks to execute, depending on the available robots and the given optimization criteria. In the current literature, there are not many approaches that attempt to generalize the planning procedure for generic tasks of class CD for heterogeneous multi-robot teams.

CHAPTER 7

Conclusion

The goal of the research presented in this thesis is to synthesize a *complete planning system for distributed task allocation, scheduling, and coordination* of heterogeneous robotic teams based on a hierarchical task representation for complex multi-robot missions. The problems considered in this work fall into the category of NP-hard problems, which are intractable for higher dimensional problem spaces. Therefore, we opt for solutions that use various approximations and heuristics to provide suboptimal solutions with fast computation times. Through intelligent problem modeling, the mission planning domain relates to a more general Vehicle Routing Problem (VRP) model. This makes the system domain-independent and can be easily used in different multi-robot applications. The main contributions of the thesis are laid out as follows.

A framework for decentralized task allocation, scheduling, and coordination of heterogeneous robotic teams based on hierarchical task representation.

The first contribution of this work is the creation of a generic framework for decentralized task allocation, scheduling, and coordination of heterogeneous robotic teams. Following Generalized Partial Global Planning (GPGP) paradigm, we have developed a system of generic modules which foster mission planning and coordination of multi-robot systems – GEneric Multirobot mission coordination and planning based on hierarchical task representation (GEM). Each module is assigned a role (task allocator, scheduler, coordinator), and all functions are implemented in ROS software packages. The implementation in the ROS environment promotes faster integration into different robotic systems and the possibility to test in the realistic hardware-in-the-loop simulation environment Gazebo. The framework includes coordination mechanisms that ensure coordinated mission planning and execution. The software infrastructure interfaces with the Task Analysis, Environment Modeling and Simulation (TÆMS) hierarchical task model.

The proposed solution was intensively tested in different application scenarios, both in simulated environments and in real-world experiments with heterogeneous teams of robots. We have shown that the framework enables timely planning, coordination, and execution of complex missions in various domains. Application scenarios include:

– *Autonomous delivery tasks using a UAV with a pair of manipulator arms cooperating with two UGVs* [97]. The approach was tested in a parcel transport scenario in a cluttered environment. We first simulated the system in a realistic Gazebo simulator and then experimentally with a 3D Robotics quadcopter and two Pioneer 3DX mobile robots. We showed that the system is capable of planning complex missions online and adapting the result to different multi-objective optimization criteria.
– *Motion symbiotic UAV-UGV team in a package delivery scenario* [100]. We propose a novel air-ground system that focuses on symbiotic behavior and extends the motion capabilities of the robotic team. To this end, we designed and constructed a lightweight UGV (L-UGV) that can work closely with a UAV equipped with a mobile manipulator (UAS) specifically designed to be transported by the UAS. We have demonstrated in simulated and experimental environments the improved efficiency of the robotic team when the robots utilize each other's capabilities to increase the energy savings and range of motion of the system.
– *Cooperative aerial-ground multi-robot system for automated construction tasks* [103]. We propose a solution for planning and coordinating the construction of a wall with a predefined structure for a heterogeneous system consisting of a UGV and up to three UAVs. We tested the method in a simulation environment on various randomized problems and compared the performance with the results obtained using an optimal solver Gurobi and a state-of-the-art auction-based method. The results show that, on average, the proposed method is within 12% of the optimum for most of the tested criteria. Although the auction-based method gives better results in terms of optimality for some examples, our method shows better scalability and robustness since we use a decentralized approach.
– *Automated greenhouse application scenario.* In this scenario, we implemented the final version of the GEM framework along with the developed allocation and scheduling algorithm for XD problems. The method is evaluated in a simulation of an automated greenhouse, where we demonstrate the ability of the method to adapt the planning strategy depending on the available robots and the given optimization criteria. We have shown that this approach is able to select the correct mission decomposition of a rather complex and highly constrained mission for different multi-objective criteria.

A method for distributed task allocation and scheduling for heterogeneous robotic team missions with cross-schedule task dependencies.

The second contribution of this thesis entails distributed task allocaiton and scheduling method for heterogeneous robotic teams [113]. Missions considered here fall in the XD[ST-MR-TA] category of mission planning problem taxonomy [38]. The contribution consists of firstly modeling of mission planning problems in terms of VRP paradigm. This modeling creates a generic mathematical model of mission planning problems which is suitable for applications in various multi-robot systems. In our solution approach, we employ multi-objective optimization with a form of distributed genetic algorithm using mimetism and knowledge sharing (CBM-pop). This approach, which uses distributed evolutionary computation methods, can quickly generate near-optimal solutions, and thus, work online while achieving good scalability properties.

In evaluating the proposed method, we generate a generic set of multi-robot task planning problems and compare our solution with Gurobi Optimizer, an optimal MILP solver, and a state-of-the-art distributed auction-based approach with a single central agent acting as an auctioneer. The results show that the proposed method performs similarly to the auction-based method in terms of optimality. For smaller problems that can be solved by the optimal solver, the obtained solutions are well within 0.5% of the optimum. However, the proposed metaheuristic exhibits much higher computational speed and scalability, which makes it more suitable for online applications to real robotic systems.

A method for distributed mission decomposition selection for heterogeneous robotic team missions with complex task dependencies.

Finally, the last contribution of this thesis tackles problems of highest complexity – CD class of problems [38]. These problems entail missions where some of the tasks may have several ways of being performed. The planning procedure needs to decide on the appropriate task decomposition given the optimization criteria and robots available in the system, and then allocate and schedule each task. Since the problem of mission decomposition selection is tightly coupled with task allocation and scheduling problems, we proposed a two-stage method that incorporates all of these components of mission planning.

The method is evaluated in a simulation setup of an automated greenhouse use case. Each of the tasks can be performed in two different ways, each requiring a different set of robots. To demonstrate the efficiency of the proposed method, we developed a large-scale use case with 20 complex tasks to be executed by three different robots. For the task decomposition selection only, the number of possible options is 2^{20}. The complexity increases further when assignments and planning permutations are added to the problem.

We have demonstrated a fast and efficient distributed method for planning complex missions for heterogeneous MRS. In the current literature, there are not many approaches that attempt to generalize the planning procedure for generic tasks of class CD for heterogeneous multi-robot teams. Therefore, this solution provides a new perspective on problems of this complexity, where this one method could be applicable to problems from different domains of MRS.

Future Work

As future work, we are interested in testing the proposed approach in a more dynamic setting and introducing protocols for handling perturbations in the system, including asynchronous and stochastic arrival of new tasks in the system. These are issues that are common in real applications and should be considered to provide a well-rounded control framework. Given the distributed nature of the proposed algorithms, we plan to consider robustness with respect to delays or information loss in the communication channel. Especially in the field operations, faulty communication networks or delays in the information can cause major problems if not taken into account. Therefore, we want to extend the proposed approach, which assumes ideal communication conditions, to include more realistic communication infrastructures. All these efforts would make the proposed solutions even more suitable for online applications in dynamic multi-robot systems.

Bibliography

[1] Yan, Z., Jouandeau, N., Cherif, A. A., “A survey and analysis of multi-robot coordination”, International Journal of Advanced Robotic Systems, Vol. 10, No. 12, 2013, page 399.

[2] Ismail, Z. H., Sariff, N., “A survey and analysis of cooperative multi-agent robot systems: Challenges and directions”, in Applications of Mobile Robots. IntechOpen, Mar. 2019.

[3] Rizk, Y., Awad, M., Tunstel, E. W., “Cooperative heterogeneous multi-robot systems: A survey”, ACM Comput. Surv., Vol. 52, No. 2, Apr. 2019.

[4] Zlot, R., Stentz, A., “Market-based multirobot coordination for complex tasks”, The International Journal of Robotics Research, Vol. 25, No. 1, 2006, pages 73-101.

[5] Garey, M. R., Johnson, D. S., Computers and Intractability; A Guide to the Theory of NP-Completeness. USA: W. H. Freeman & Co., 1990.

[6] Dahl, T. S., Matarić, M., Sukhatme, G. S., “Multi-robot task allocation through vacancy chain scheduling”, Robotics and Autonomous Systems, Vol. 57, No. 6-7, Jun. 2009, pages 674–687.

[7] Wawerla, J., Vaughan, R. T., “A fast and frugal method for team-task allocation in a multi-robot transportation system”, in 2010 IEEE International Conference on Robotics and Automation (ICRA). IEEE, May 2010.

[8] Bicchi, A., Danesi, A., Dini, G., Porta, S., Pallottino, L., Savino, I., Schiavi, R., “Heterogeneous wireless multirobot system”, IEEE Robotics & Automation Magazine, Vol. 15, No. 1, Mar. 2008, pages 62–70.

[9] Pei, Y., Mutka, M. W., “Steiner traveler: Relay deployment for remote sensing in heterogeneous multi-robot exploration”, in 2012 IEEE International Conference on Robotics and Automation (ICRA). IEEE, May 2012.

[10] Farinelli, A., Iocchi, L., Nardi, D., “Multirobot systems: A classification focused on coordination”, IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society, Vol. 34, Nov. 2004, pages 2015-28.

[11] Zhang, L., Wong, T., “An object-coding genetic algorithm for integrated process planning and scheduling”, European Journal of Operational Research, Vol. 244, No. 2, Jul. 2015, pages 434–444.

[12] Gombolay, M., Wilcox, R., Shah, J., “Fast Scheduling of Multi-Robot Teams with Temporospatial Constraints”, IEEE Transactions on Robotics (T-RO), 2018, pages 1–20.

[13] García, P., Caamaño, P., Duro, R. J., Bellas, F., “Scalable task assignment for heterogeneous multi-robot teams”, International Journal of Advanced Robotic Systems, Vol. 10, No. 2, Feb. 2013, page 105.

[14] Korsah, G. A., “Exploring Bounded Optimal Coordination for Heterogeneous Teams with Cross-Schedule Dependencies”, PhD thesis, Carnegie Mellon University, Jan. 2011.

[15] Gerkey, B., Mataric, M., “Sold!: auction methods for multirobot coordination”, IEEE Transactions on Robotics and Automation, Vol. 18, No. 5, Oct. 2002, pages 758–768.

[16] Cortes, J., Egerstedt, M., “Coordinated control of multi-robot systems: A survey”, SICE Journal of Control, Measurement, and System Integration, Vol. 10, No. 6, 2017, pages 495-503.

[17] Schillinger, P., Bürger, M., Dimarogonas, D. V., “Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems”, Vol. 37, No. 7, 2018, pages 818-838.

[18] Srinivasan, M., Coogan, S., Egerstedt, M., “Control of multi-agent systems with finite time control barrier certificates and temporal logic”, in 2018 IEEE Conference on Decision and Control (CDC), 2018, pages 1991-1996.

[19] DeCastro, J. A., Alonso-Mora, J., Raman, V., Rus, D., Kress-Gazit, H., “Collision-Free Reactive Mission and Motion Planning for Multi-robot Systems”, in Springer Proceedings in Advanced Robotics. Springer, Cham, 2018, pages 459–476.

[20] Kantaros, Y., Malencia, M., Kumar, V., Pappas, G. J., “Reactive temporal logic planning for multiple robots in unknown environments”, in 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pages 11 479-11 485.

[21] Poole, D. L., Mackworth, A. K., Artificial Intelligence: foundations of computational agents. Cambridge University Press, 2010.

[22] Kantaros, Y., Zavlanos, M. M., “Stylus*: A temporal logic optimal control synthesis algorithm for large-scale multi-robot systems”, The International Journal of Robotics Research, Vol. 39, No. 7, 2020, pages 812-836.

[23] Morere, P., Marchant, R., Ramos, F., “Sequential Bayesian optimization as a POMDP for environment monitoring with UAVs”, in 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017, pages 6381-6388.

[24] Pajarinen, J., Kyrki, V., “Robotic manipulation of multiple objects as a POMDP”, Artificial Intelligence, Vol. 247, 2017, pages 213-228, special Issue on AI and Robotics.

[25] Amato, C., “Decision-making under uncertainty in multi-agent and multi-robot systems: Planning and learning”, in Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18. International Joint Conferences on Artificial Intelligence Organization, July 2018, pages 5662–5666.

[26] Floriano, B., Borges, G. A., Ferreira, H., “Planning for Decentralized Formation Flight of UAV fleets in Uncertain Environments with Dec-POMDP”, in 2019 International Conference on Unmanned Aircraft Systems (ICUAS). IEEE, Jun. 2019, pages 563–568.

[27] Garg, N. P., Hsu, D., Lee, W. S., “DESPOT-Alpha: Online POMDP planning with large state and observation spaces”, in Robotics: Science and Systems XV, University of Freiburg, Freiburg im Breisgau, Germany, Jun. 22-26, 2019, Bicchi, A., Kress-Gazit, H., Hutchinson, S., (ur.), 2019.

[28] Capitan, J., Spaan, M. T., Merino, L., Ollero, A., “Decentralized multi-robot cooperation with auctioned POMDPs”, in 2012 IEEE International Conference on Robotics and Automation. IEEE, May 2012.

[29] Liu, M., Sivakumar, K., Omidshafiei, S., Amato, C., How, J. P., “Learning for multi-robot cooperation in partially observable stochastic environments with macro-actions”, in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pages 1853-1860.

[30] Beni, G., Wang, J., “Swarm intelligence in cellular robotic systems”, in Robots and Biological Systems: Towards a New Bionics?, Dario, P., Sandini, G., Aebischer, P., (ur.). Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pages 703–712.

[31] Brambilla, M., Ferrante, E., Birattari, M., Dorigo, M., “Swarm robotics: A review from the swarm engineering perspective”, Swarm Intelligence, Vol. 7, Mar. 2013, pages 1-41.

[32] Dorigo, M., Floreano, D., Gambardella, L., Mondada, F., Nolfi, S., Baaboura, T., Birattari, M., Bonani, M., Brambilla, M., Brutschy, A., Burnier, D., Campo, A., Christensen, A., Decugniere, A., Di Caro, G., Ducatelle, F., Ferrante, E., Forster, A., Gonzales, J., Guzzi, J., Longchamp, V., Magnenat, S., Mathews, N., Montes de Oca, M., O’Grady, R., Pinciroli, C., Pini, G., Retornaz, P., Roberts, J., Sperati, V., Stirling, T., Stranieri, A., Stutzle, T., Trianni, V., Tuci, E., Turgut, A., Vaussard, F., “Swarmanoid: A novel concept for the study of heterogeneous robotic swarms”, IEEE Robotics and Automation Magazine, Vol. 20, No. 4, 2013, pages 60–71.

[33] Kennedy, J., Eberhart, R., “Particle swarm optimization”, in Proceedings of ICNN’95 - International Conference on Neural Networks, Vol. 4, 1995, pages 1942-1948 vol.4.

[34] Duan, H., Qiao, P., “Pigeon-inspired optimization: a new swarm intelligence optimizer for air robot path planning”, International Journal of Intelligent Computing and Cybernetics, 2014.

[35] Teodorovic, D., Lucic, P., Markovic, G., Dell’Orco, M., “Bee colony optimization: principles and applications”, in 2006 8th Seminar on Neural Network Applications in Electrical Engineering. IEEE, 2006, pages 151–156.

[36] Zlot, R. M., “An auction-based approach to complex task allocation for multirobot teams”, PhD thesis, Carnegie Mellon University, 2006.

[37] Gerkey, B. P., Matarić, M. J., “A formal analysis and taxonomy of task allocation in multi-robot systems”, The International journal of robotics research, Vol. 23, No. 9, 2004, pages 939–954.

[38] Korsah, G. A., Stentz, A., Dias, M. B., “A comprehensive taxonomy for multi-robot

task allocation", The International Journal of Robotics Research, Vol. 32, No. 12, 2013, pages 1495-1512.

[39] Nunes, E., Manner, M., Mitiche, H., Gini, M., "A taxonomy for task allocation problems with temporal and ordering constraints", Robotics and Autonomous Systems, Vol. 90, 2017, pages 55–70.

[40] Koes, M., Nourbakhsh, I., Sycara, K., "Heterogeneous multirobot coordination with spatial and temporal constraints", in Proceedings of the 20th National Conference on Artificial Intelligence - Volume 3, Vol. 5, 2005, pages 1292–1297.

[41] Ramchurn, S. D., Polukarov, M., Farinelli, A., Jennings, N., Trong, C., "Coalition formation with spatial and temporal constraints", in International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), 2010, pages 1181–1188.

[42] Alighanbari, M., Kuwata, Y., How, J. P., "Coordination and control of multiple UAVs with timing constraints and loitering", in Proceedings of the 2003 American Control Conference, 2003., Vol. 6. IEEE, 2003, pages 5311–5316.

[43] International Business Machines Corporation (IBM), "CPLEX Optimizer", available at: https://www.ibm.com/analytics/cplex-optimizer Accessed: 2021-09-20.

[44] Gurobi Optimization, LLC, "Gurobi optimizer reference manual", available at: http://www.gurobi.com Accessed: 2021-09-20. 2019.

[45] Jünger, M., Thienel, S., "The ABACUS system for branch-and-cut-and-price algorithms in integer programming and combinatorial optimization", Software Practice and Experience, Vol. 30, September 2000, pages 1325-1352.

[46] MIT - Massachusetts Institute of Technology, "lp_solve optimizer", available at: http://web.mit.edu/lpsolve/doc/ Accessed: 2021-09-20.

[47] Mitten, L., "Branch-and-bound methods: General formulation and properties", Operations Research, Vol. 18, No. 1, 1970, pages 24–34.

[48] Korsah, G. A., Kannan, B., Browning, B., Stentz, A., Dias, M. B., "xBots: An approach to generating and executing optimal multi-robot plans with cross-schedule dependencies", in 2012 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2012, pages 115–122.

[49] Dokeroglu, T., Sevinc, E., Kucukyilmaz, T., Cosar, A., "A survey on new generation metaheuristic algorithms", Computers & Industrial Engineering, Vol. 137, 2019, page 106040.

[50] Wei, C., Ji, Z., Cai, B., "Particle swarm optimization for cooperative multi-robot task allocation: a multi-objective approach", IEEE Robotics and Automation Letters, Vol. 5, No. 2, 2020, pages 2530–2537.

[51] Chen, X., Zhang, P., Du, G., Li, F., "Ant colony optimization based memetic algorithm to solve bi-objective multiple traveling salesmen problem for multi-robot systems", IEEE Access, Vol. 6, 2018, pages 21 745–21 757.

[52] Çakar, T., Köker, R., Demir, H. I., "Parallel robot scheduling to minimize mean tardiness with precedence constraints using a genetic algorithm", Advances in Engineering Software, Vol. 39, No. 1, 2008, pages 47–54.

[53] Mitiche, H., Boughaci, D., Gini, M., "Iterated local search for time-extended multi-robot task allocation with spatio-temporal and capacity constraints", Journal of Intelligent Systems, Vol. 28, No. 2, 2019, pages 347–360.

[54] Gini, M., "Multi-robot allocation of tasks with temporal and ordering constraints", in Thirty-First AAAI Conference on Artificial Intelligence, 2017.

[55] Nunes, E., McIntire, M., Gini, M., "Decentralized multi-robot allocation of tasks

with temporal and precedence constraints", Advanced Robotics, Vol. 31, No. 22, Nov. 2017, pages 1193–1207.

[56] Lemaire, T., Alami, R., Lacroix, S., "A distributed tasks allocation scheme in multi-UAV context", in in Proceedings of IEEE International Conference on Robotics and Automation (ICRA), Vol. 4. IEEE, 2004, pages 3622–3627.

[57] Choi, H.-L., Brunet, L., How, J. P., "Consensus-based decentralized auctions for robust task allocation", IEEE Transactions on Robotics (T-RO), Vol. 25, No. 4, 2009, pages 912–926.

[58] Godoy, J., Gini, M., "Task allocation for spatially and temporally distributed tasks", in Intelligent Autonomous Systems 12. Springer, 2013, pages 603–612.

[59] Motes, J., Sandström, R., Lee, H., Thomas, S., Amato, N. M., "Multi-robot task and motion planning with subtask dependencies", IEEE Robotics and Automation Letters, Vol. 5, No. 2, 2020, pages 3338-3345.

[60] Jones, E. G., Dias, M. B., Stentz, A., "Time-extended multi-robot coordination for domains with intra-path constraints", Autonomous robots, Vol. 30, No. 1, 2011, pages 41–56.

[61] Horling, B., Lesser, V., Vincent, R., Wagner, T., Raja, A., Zhang, S., Decker, K., Garvey, A., "The TAEMS white paper", available at: http://mas.cs.umass.edu/paper/182 1999.

[62] Lesser, V., Decker, K., Wagner, T., Carver, N., Garvey, A., Horling, B., Neiman, D., Podorozhny, R., Prasad, M. N., Raja, A., Vincent, R., Xuan, P., Zhang, X. Q., "Evolution of the GPGP/TÆMS domain-independent coordination framework", Autonomous Agents and Multi-Agent Systems, Vol. 9, No. 1, July 2004, pages 87–143.

[63] Wagner, T., Garvey, A., Lesser, V., "Criteria-directed task scheduling", International Journal of Approximate Reasoning, Vol. 19, No. 1, 1998, pages 91–118.

[64] Decker, K. S., Lesser, V. R., "Designing a family of coordination algorithms", in Proceedings of the First International Conference on Multi-Agent System. AAAI Press, 1995, pages 73–80.

[65] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A. Y., "ROS: an open-source Robot Operating System", in ICRA workshop on open source software, Vol. 3, No. 3.2, 2009, page 5.

[66] Toth, P., Vigo, D., for Industrial, S., Mathematics, A., Vehicle Routing: Problems, Methods, and Applications, ser. MOS-SIAM series on optimization. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2015.

[67] El-Sherbeny, N. A., "Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods", Journal of King Saud University - Science, Vol. 22, No. 3, 2010, pages 123-131.

[68] Ait Haddadene, S. R., Labadie, N., Prodhon, C., "A GRASP x ILS for the vehicle routing problem with time windows, synchronization and precedence constraints", Expert Systems with Applications, Vol. 66, 2016, pages 274-294.

[69] Ramos, T. R. P., Gomes, M. I., Póvoa, A. P. B., "Multi-depot vehicle routing problem: a comparative study of alternative formulations", International Journal of Logistics Research and Applications, Vol. 23, No. 2, 2020, pages 103-120.

[70] Baldacci, R., Battarra, M., Vigo, D., Routing a Heterogeneous Fleet of Vehicles, Jan. 2008, Vol. 43, pages 3-27.

[71] Li, F., Golden, B., Wasil, E., “The open vehicle routing problem: Algorithms, large-scale test problems, and computational results”, Computers & Operations Research, Vol. 34, No. 10, 2007, pages 2918 - 2930.

[72] Laporte, G., Nobert, Y., D, A., “Optimal solutions to capacitated multidepot vehicle routing problems”, Congressus Numerantium, Vol. 44, 1984, pages 283–292.

[73] Laporte, G., Norbert, Y., Taillefer, S., “Solving a family of multi-depot vehicle routing and location-routing problems”, Transportation Science, Vol. 22, No. 3, 1988, pages 161–172.

[74] Baldacci, R., Mingozzi, A., “An unified exact method for solving different classes of vehicle routing problems”, Mathematical Programming, Vol. 120, Spetember 2009, pages 347-380.

[75] Contardo, C., Martinelli, R., “A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints”, Discrete Optimization, Vol. 12, 2014, pages 129 - 146.

[76] Montoya-Torres, J. R., López Franco, J., Nieto Isaza, S., Felizzola Jiménez, H., Herazo-Padilla, N., “A literature review on the vehicle routing problem with multiple depots”, Computers & Industrial Engineering, Vol. 79, 2015, pages 115-129.

[77] Abdel-Basset, M., Abdel-Fatah, L., Sangaiah, A. K., “Chapter 10 - metaheuristic algorithms: A comprehensive review”, in Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications, ser. Intelligent Data-Centric Systems, Sangaiah, A. K., Sheng, M., Zhang, Z., (ur.). Academic Press, 2018, pages 185 - 231.

[78] Mirjalili, S., “Genetic algorithm”, in Evolutionary algorithms and neural networks. Springer, 2019, pages 43–55.

[79] Ombuki-Berman, B., Hanshar, F., “Using genetic algorithms for multi-depot vehicle routing”, Studies in Computational Intelligence, Vol. 161, September 2008, pages 77-99.

[80] Gesú, V., Giancarlo, R., Lo Bosco, G., Raimondi, A., Scaturro, D., “Genclust: A genetic algorithm for clustering gene expression data”, BMC bioinformatics, Vol. 6, Feb. 2005, page 289.

[81] Mahmud, N., Haque, M. M., “Solving multiple depot vehicle routing problem (MDVRP) using genetic algorithm”, in 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), 2019, pages 1-6.

[82] Venkata Narasimha, K., Kivelevitch, E., Sharma, B., Kumar, M., “An ant colony optimization technique for solving min–max multi-depot vehicle routing problem”, Swarm and Evolutionary Computation, Vol. 13, 2013, pages 63 - 73.

[83] Zhang, S., Zhang, W., Gajpal, Y., Appadoo, S. S., Ant Colony Algorithm for Routing Alternate Fuel Vehicles in Multi-depot Vehicle Routing Problem. Springer Singapore, 2019, pages 251–260.

[84] Escobar, J., Linfati, R., Toth, P., Baldoquin, M., “A hybrid granular tabu search algorithm for the multi-depot vehicle routing problem”, Journal of Heuristics, Vol. 20, Oct. 2014, pages 1-27.

[85] Luo, J., Chen, M.-R., “Multi-phase modified shuffled frog leaping algorithm with extremal optimization for the MDVRP and the MDVRPTW”, Computers & Industrial Engineering, Vol. 72, 2014, pages 84 - 97.

[86] Mirabi, M., Fatemi Ghomi, S., Jolai, F., “Efficient stochastic hybrid heuristics for the multi-depot vehicle routing problem”, Robotics and Computer-Integrated

Manufacturing, Vol. 26, No. 6, 2010, pages 564 - 569, 19th International Conference on Flexible Automation and Intelligent Manufacturing.

[87] Haerani, E., Wardhani, L. K., Putri, D. K., Sukmana, H. T., “Optimization of multiple depot vehicle routing problem (MDVRP) on perishable product distribution by using genetic algorithm and fuzzy logic controller (FLC)”, in 2017 5th International Conference on Cyber and IT Service Management (CITSM), 2017, pages 1-5.

[88] Stodola, P., “Using metaheuristics on the multi-depot vehicle routing problem with modified optimization criterion”, Algorithms, Vol. 11, No. 5, 2018.

[89] de Oliveira, F. B., Enayatifar, R., Sadaei, H. J., Guimarães, F. G., Potvin, J.-Y., “A cooperative coevolutionary algorithm for the multi-depot vehicle routing problem”, Expert Systems with Applications, Vol. 43, 2016, pages 117-130.

[90] Sadati, M. E. H., Aksen, D., Aras, N., “The r-interdiction selective multi-depot vehicle routing problem”, International Transactions in Operational Research, Vol. 27, No. 2, 2020, pages 835-866.

[91] Meignan, D., Koukam, A., Créput, J.-C., “Coalition-based metaheuristic: A self-adaptive metaheuristic using reinforcement learning and mimetism”, Journal of Heuristics, Vol. 16, Dec. 2010, pages 859-879.

[92] Kennedy, J., Eberhart, R., “Particle swarm optimization”, in Proceedings of ICNN’95 - International Conference on Neural Networks, Vol. 4, 1995, pages 1942-1948 vol.4.

[93] Sprecher, A., Kolisch, R., Drexl, A., “Semi-active, active, and non-delay schedules for the resource-constrained project scheduling problem”, European Journal of Operational Research, Vol. 80, No. 1, 1995, pages 94-102.

[94] Chang, K.-H., “Chapter 19 - multiobjective optimization and advanced topics”, in e-Design, Chang, K.-H., (ur.). Boston: Academic Press, 2015, pages 1105 - 1173.

[95] Deb, K., Kalyanmoy, D., Multi-Objective Optimization Using Evolutionary Algorithms. USA: John Wiley & Sons, Inc., 2001.

[96] Baptista Pereira, F., Tavares, J., (ur.), Bio-inspired Algorithms for the Vehicle Routing Problem, 1st ed. Springer-Verlag Berlin Heidelberg, 2009, Vol. 161.

[97] Arbanas, B., Ivanovic, A., Car, M., Haus, T., Orsag, M., Petrovic, T., Bogdan, S., “Aerial-ground robotic system for autonomous delivery tasks”, in 2016 IEEE International Conference on Robotics and Automation (ICRA). IEEE, May 2016.

[98] Petrovic, T., Haus, T., Arbanas, B., Orsag, M., Bogdan, S., “Can UAV and UGV be best buddies? towards heterogeneous aerial-ground cooperative robot system for complex aerial manipulation tasks”, in 2015 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO), Vol. 01, July 2015, pages 238-245.

[99] LARICS lab, “Experiment video - aerial ground robotic system for autonomous delivery tasks”, https://www.youtube.com/watch?v=LPQFzzy7bnM, accessed: 2021-09-20.

[100] Arbanas, B., Ivanovic, A., Car, M., Orsag, M., Petrovic, T., Bogdan, S., “Decentralized planning and control for UAV–UGV cooperative teams”, Autonomous Robots, Feb. 2018.

[101] Cvisic, I., Petrovic, I., “Stereo odometry based on careful feature selection and tracking”, in 2015 European Conference on Mobile Robots (ECMR), September 2015, pages 1-6.

[102] LARICS lab, “Experiment video - Decentralized planning and control for UAV–UGV cooperative teams”, https://goo.gl/0J1hmK, accessed: 2021-09-20.

[103] Krizmancic, M., Arbanas, B., Petrovic, T., Petric, F., Bogdan, S., “Cooperative aerial-ground multi-robot system for automated construction tasks”, IEEE Robotics and Automation Letters, Vol. 5, No. 2, Apr. 2020, pages 798–805.

[104] MBZIRC, “The MBZIRC Challenge”, available at: https://www.mbzirc.com Accessed: 2021-09-20.

[105] Kundakcioglu, O. E., Alizamir, S., “Generalized assignment problem”, in Encyclopedia of Optimization, Floudas, C. A., Pardalos, P. M., (ur.). Boston, MA: Springer US, 2009, pages 1153–1162.

[106] Ross, G. T., Soland, R. M., “A branch and bound algorithm for the generalized assignment problem”, Math. Program., Vol. 8, No. 1, Dec. 1975, page 91–103.

[107] LARICS lab, “Simulation Video - Cooperative Aerial-Ground Multi-Robot System for Automated Construction Tasks”, available at: https://www.youtube.com/watch?v=-0DU3w8jWWA Accessed: 2021-09-20.

[108] LARICS lab, “Additional Results - Cooperative Aerial-Ground Multi-Robot System for Automated Construction Tasks”, available at: http://larics.github.io/GPGP_page Accessed: 2021-09-20.

[109] E.T.S.I. Informática, U. o. M., “Cordeau Multiple Depot VRP Instances”, https://neo.lcc.uma.es/vrp/vrp-instances/multiple-depot-vrp-instances/, accessed: 2021-09-20.

[110] Cordeau, J.-F., Gendreau, M., Laporte, G., “A tabu search heuristic for periodic and multi-depot vehicle routing problems”, Networks, Vol. 30, No. 2, 1997, pages 105-119.

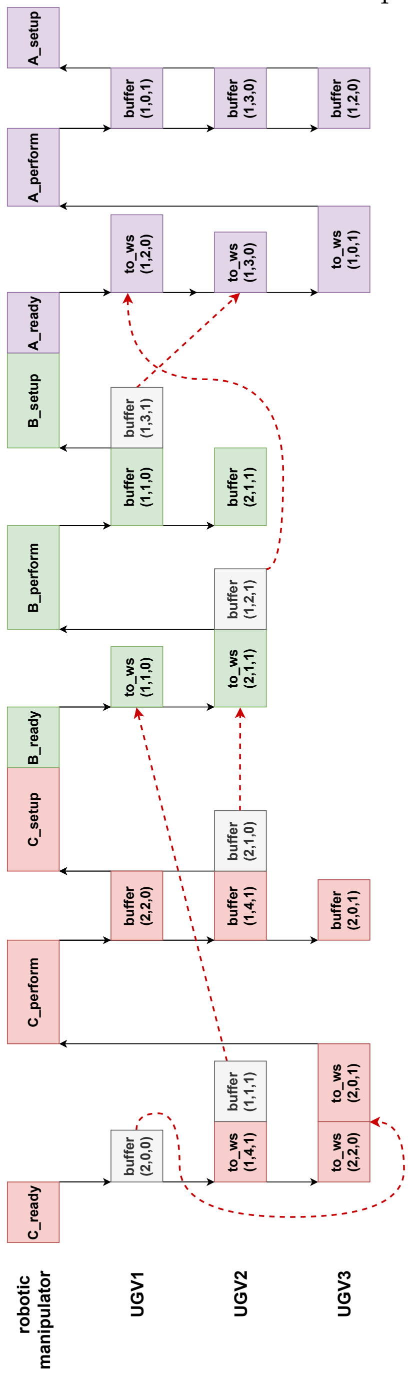
[111] Arbanas, B., Petrovic, T., Orsag, M., Martínez-de Dios, J. R., Bogdan, S., “Additional results - distributed allocation and scheduling of tasks with cross-schedule dependencies for heterogeneous multi-robot teams”, https://sites.google.com/view/vrp-task-planning/home, accessed: 2020-11-17. 2021.

[112] LARICS lab. Task planning benchmark repository. https://sites.google.com/view/taskplanningrepository/home. Accessed on 2021-09-20.

[113] Ferreira, B. A., Petrović, T., Orsag, M., de Dios, J. R. M., Bogdan, S., “Distributed allocation and scheduling of tasks with cross-schedule dependencies for heterogeneous multi-robot teams”, available at: https://arxiv.org/abs/2109.03089 2021.

Appendix

Appendix A. SpECULARIA use case schedule for XD problems



Acronyms

CBM	Coalition-Based Metaheuristic	57
CVRP	Capacitated Vehicle Routing Problem	39
DTC	Design-To-Criteria planner	24
GEM	GEneric Multirobot mission coordination and planning based on hierarchical task representation	119
GPGP	Generalized Partial Global Planning	23, 86, 119
HFVRP	Heterogeneous Fleet Vehicle Routing Problem	46
HFVRP-PS	Heterogeneous Fleet Vehicle Routing Problem with Time Windows and Precedence and Synchronization Constraints	60
MDVRP	Multi-Depot Vehicle Routing Problem	44, 57, 99
MILP	Mixed-Integer Linear Programming	14, 40, 95
MRS	Multi-Robot Systems	1, 5
OMPL	Open Motion Planning Library	73
POMDPs	Partially Observable Markov Decision Processes	9
QAF	Quality Accumulation Function	20, 90
ROS	Robot Operating System	31, 92
TÆMS	Task Analysis, Environment Modeling and Simulation	2, 18, 119
UAS	Unmanned Aerial System	77
UAV	Unmanned Aerial Vehicle	71
UGV	Unmanned Ground Vehicle	71
VRP	Vehicle Routing Problem	2, 4, 14, 33, 119, 121
VRPTW	Vehicle Routing Problem with Time Windows	41
VRPTW-PS	Vehicle Routing Problem with Time Windows and Precedence and Synchronization Constraints	42

List of Figures

List of Tables

Biography

Barbara Arbanas Pascoal Ferreira received her diploma in Computing in 2015 from the Faculty of Electrical Engineering and Computing University in Zagreb, majoring in Computer Science. Since then, she has been employed at the Laboratory for Robotics and Intelligent Control Systems (LARICS) under the supervision of Prof. Stjepan Bogdan, where she has been working as a research and teaching assistant. Coming from computer science background, her interests include multi-robot coordination and planning, distributed artificial intelligence, scheduling, and optimization. As a researcher, she has been involved in several international and national research projects, including H2020 project subCULTron, FP7 EuRoC, and Croatian Science Foundation project SpECULARIA. In addition, she participated as a member of the LARICS team in ERL Emergency Robots 2019 and MBZIRC 2020 robotics competitions. So far, she has authored or co-authored 8 conference papers, 5 journal papers, and one book chapter. She is a student IEEE member and an active member of the IEEE OES UNIZG Student Branch Chapter.

List of publications (Barbara Arbanas Pascoal Ferreira)

Book chapters

[1] Arbanas, B., Petric, F., Batinović, A., Polić, M., Vatavuk, I., Marković, L., Car, M., Hrabar, I., Ivanović, A., Bogdan, S., “From ERL to MBZIRC: Development of an aerial-ground robotic team for search and rescue”, 2021.

Journal papers

[1] Arbanas, B., Ivanovic, A., Car, M., Orsag, M., Petrovic, T., Bogdan, S., “Decentralized planning and control for UAV–UGV cooperative teams”, Autonomous Robots, February 2018.

[2] Lončar, I., Babić, A., Arbanas, B., Vasiljević, G., Petrović, T., Bogdan, S., Mišković, N., “A heterogeneous robotic swarm for long-term monitoring of marine environments”, Applied Sciences, Vol. 9, No. 7, 2019.

[3] Krizmancic, M., Arbanas, B., Petrovic, T., Petric, F., Bogdan, S., “Cooperative aerial-ground multi-robot system for automated construction tasks”, IEEE Robotics and Automation Letters, Vol. 5, No. 2, Apr. 2020, pages 798–805.

[4] Babić, A., Lončar, I., Arbanas, B., Vasiljević, G., Petrović, T., Bogdan, S., Mišković, N., “A novel paradigm for underwater monitoring using mobile sensor networks”, Sensors, Vol. 20, No. 16, 2020.

[5] Vasiljević, G., Petrović, T., Arbanas, B., Bogdan, S., "Dynamic median consensus for marine multi-robot systems using acoustic communication", IEEE Robotics and Automation Letters, Vol. 5, No. 4, Oct 2020, pages 5299-5306.

Conference papers

[1] Petrovic, T., Haus, T., Arbanas, B., Orsag, M., Bogdan, S., "Can UAV and UGV be best buddies? towards heterogeneous aerial-ground cooperative robot system for complex aerial manipulation tasks", in 2015 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO), Vol. 01, July 2015, pages 238-245.

[2] Arbanas, B., Ivanovic, A., Car, M., Haus, T., Orsag, M., Petrovic, T., Bogdan, S., "Aerial-ground robotic system for autonomous delivery tasks", in 2016 IEEE International Conference on Robotics and Automation (ICRA). IEEE, May 2016.

[3] Mazdin, P., Arbanas, B., Haus, T., Bogdan, S., Petrovic, T., Miskovic, N., "Trust consensus protocol for heterogeneous underwater robotic systems", IFAC-PapersOnLine, Vol. 49, No. 23, 2016, pages 341-346, 10th IFAC Conference on Control Applications in Marine Systems CAMS 2016.

[4] Arbanas, B., Boljuncic, S., Petrovic, T., Bogdan, S., "Decentralized energy sharing protocol using TÆMS framework and coalition-based metaheuristic for heterogeneous robotic systems", IFAC-PapersOnLine, Vol. 50, No. 1, 2017, pages 5914-5919, 20th IFAC World Congress.

[5] Arbanas, B., Petrovic, T., Bogdan, S., "Consensus protocol for underwater multi-robot system using scheduled acoustic communication", in 2018 OCEANS - MTS/IEEE Kobe Techno-Oceans (OTO), 2018, pages 1-5.

[6] Arbanas, B., Petrovic, T., Bogdan, S., "Consensus protocol for underwater multi-robot system using two communication channels", in 2018 26th Mediterranean Conference on Control and Automation (MED), 2018, pages 358-363.

[7] Vasiljevic, G., Arbanas, B., Bogdan, S., "Ambient light based depth control of underwater robotic unit aMussel", in 2019 International Conference on Robotics and Automation (ICRA), 2019, pages 4640-4645.

[8] Polic, M., Ivanovic, A., Maric, B., Arbanas, B., Tabak, J., Orsag, M., "Structured ecological cultivation with autonomous robots in indoor agriculture", in 2021 16th International Conference on Telecommunications (ConTEL). IEEE, 2021, pages 189–195.

Životopis

Barbara Arbanas Pascoal Ferreira završila je 2015. godine diplomski studij računarstva Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu. Po završetku studija zaposlila se na istom Fakultetu u Laboratoriju za robotiku i inteligentne sustave upravljanjanja pod mentorstvom prof. dr. sc. Stjepana Bogdana, gdje od tada radi kao znanstvena suradnica i asistentica u nastavi. Njezini interesi uključuju koordinaciju i planiranje misija višerobotskih sustava, umjetnu inteligenciju, raspoređivanje i optimizaciju. Kao istraživač, bila je uključena u nekoliko međunarodnih i nacionalnih istraživačkih projekata, uključujući projekt H2020 subCULTron, projekt FP7 EuRoC i projekt Hrvatske zaklade za znanost SpECULARIA. Uz to, sudjelovala je kao član LARICS tima na ERL Emergency Robots 2019 i MBZIRC 2020 robotičkim natjecanjima. Autorica je ili koautorica 8 radova na internacionalnim konferencijama, 5 članaka u znanstvenim časopisima i jednog poglavlja u knjizi. Studentski je član IEEE udruge profesionalnih inženjera i aktivni član studentskog ogranka IEEE OES UNIZG.