A hyper-heuristic approach to achieving long-term autonomy in a heterogeneous swarm of marine robots

Babić, Anja

University of Zagreb

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Anja Babić

A hyper-heuristic approach to achieving long-term autonomy in a heterogeneous swarm of marine robots

DOCTORAL THESIS

Zagreb, 2023

University of Zagreb

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Anja Babić

A hyper-heuristic approach to achieving long-term autonomy in a heterogeneous swarm of marine robots

DOCTORAL THESIS

Supervisor: Professor Nikola Mišković, PhD

Zagreb, 2023

Sveučilište u Zagrebu
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Anja Babić

Hiperheuristički pristup ostvarivanju dugoročne autonomije u heterogenome roju pomorskih robota

DOKTORSKI RAD

Mentor: prof. dr. sc. Nikola Mišković

Zagreb, 2023.

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

Supervisor: Professor Nikola Mišković, PhD

DOCTORAL THESIS has: 142 pages

Dissertation No.: _____

About the Supervisor

Nikola Mišković is a Full Professor at University of Zagreb, Faculty of Electrical Engineering and Computing where he teaches control engineering related courses. He received his diploma and Ph.D. degrees in electrical engineering from the University of Zagreb, Faculty of Electrical Engineering and Computing (FER), Zagreb, Croatia, in 2005 and 2010, respectively.

From July 2005 he is working at the Department of control and computer engineering at FER. He was a visiting researcher at the Consiglio Nazionale delle Ricerche in Genoa, Italy (in 2008). In 2016 he was promoted to Associate Professor, and became Full Professor in 2019. He is the Head of Laboratory for Underwater Systems and Technologies (LABUST). He participated in 15 European projects (Horizon Europe, H2020, FP7, DG-ECHO, INTERREG) out of which he coordinated FP7 CADDY, focussing on the development of the first underwater robot for interaction with divers; H2020 aPad, devoted to commercialization of an autonomous surface vehicles developed in LABUST, and H2020 EXCELLABUST devoted to increasing LABUST excellence in marine robotics. He also participated in 4 Office of Naval Research Global (ONR-G) projects (coordinated 3), 2 NATO projects, and 7 national projects (coordinated 3). He published more than 70 papers in journals and conference proceedings in the area of navigation, guidance and control, as well as cooperative control in marine robotics.

Prof. Mišković is a member of IEEE (president of Chapter for Robotics and Automation of the Croatian Section from 2016 to 2017), IFAC (member of the Technical Committee on Marine Systems) and Centre for Underwater Systems and Technologies (vice-president since 2010).

In 2020 Prof Nikola Mišković was awarded IEEE Croatia Section Award for Outstanding Engineering Contribution for exceptional engineering contribution in the field of marine robotics, particularly innovative underwater robotic systems and autonomous surface vehicles. He received the annual State science award for 2015, awarded by the Croatian Parliament and in 2013 he received the young scientist award "Vera Johanides" of the Croatian Academy of Engineering (HATZ) for scientific achievements.

O mentoru

Nikola Mišković redoviti je profesor na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu gdje predaje kolegije vezane uz automatiku i automatsko upravljanje. Diplomirao je i doktorirao u polju elektrotehnike na Sveučilištu u Zagrebu Fakultetu elektrotehnike i računarstva (FER) 2005. odnosno 2010. godine.

Od srpnja 2005. godine radi na Zavodu za automatiku i računalno inženjerstvo FER-a. Bio je gostujući istraživač na Consiglio Nazionale delle Ricerche u Genovi, Italija (u 2008. godini). 2016. godine izabran je u zvanje izvanrednog profesora, a 2019. u zvanje redovitog profesora. Voditelj je Laboratorija za podvodne sustave i tehnologije (LAPOST). Sudjelovao je u 15 europskih projekata (Horizon Europe, H2020, FP7, DG-ECHO, INTERREG) od kojih je koordinirao FP7 CADDY, projekt fokusiran na razvoj prvog podvodnog robota za interakciju s roniocima; H2020 aPad, posvećen komercijalizaciji autonomnih površinskih vozila razvijenih u LABUST-u, i H2020 EXCELLABUST posvećen povećanju LABUST-ove izvrsnosti u pomorskoj robotici. Također je sudjelovao u 4 projekta Office of Naval Research Global (ONR-G) (od kojih 3 kao koordinator), 2 NATO projekta i 7 nacionalnih projekata (koordinator 3). Objavio je više od 70 radova u časopisima i zbornicima konferencija u području navigacije, vođenja i upravljanja, te kooperativnog upravljanja u pomorskoj robotici.

Prof. Mišković član je stručnih udruga IEEE (predsjednik Odjela za robotiku i automatizaciju Hrvatske sekcije od 2016. do 2017.), IFAC (član Technical Committee on Marine Systems) i Centra za podvodne sustave i tehnologije (dopredsjednik društva od 2010.).

Godine 2020. dobitnik je Nagrade Hrvatske sekcije IEEE za izniman inženjerski doprinos (IEEE Croatia Section Outstanding Engineer Award) za izniman inženjerski doprinos u području pomorske robotike, posebice inovativnih podvodnih robotskih sustava i autonomnih površinskih vozila. Godine 2013. primio je nagradu Hrvatske akademije tehničkih znanosti „Vera Johanides" mladom znanstveniku za uspjehe u području istraživanja, te je nagrađen godišnjom Državnom nagradom za znanost za 2015. godinu.

...

Acknowledgements

I would like to thank my advisor Prof. Nikola Mišković for always encouraging curiosity and initiative. Thanks to all my colleagues at LABUST, without whom none of the numerous field trials that led to so many of the results in this thesis would have been possible. And last but certainly not least, thanks to my family and my friends for their enduring patience and support, and to Kruno for everything from late-night brainstorming to a near-endless supply of tea.

Abstract

For the purpose of enabling long-term autonomy of a heterogeneous swarm of marine robots, task allocation and sequencing were introduced into the system's energy management procedures and agent interactions. In a scenario where the system needs to autonomously go about its monitoring mission and survive long-term, the available maximum capacity of five surface vehicles - aPad platforms which represent the charging hubs of the system - is usually outnumbered by the number of active charging requests by the sensor node-like aMussel agents, leading to a need for careful planning and optimisation of robot activities.

In the scope of this thesis, a two-layered system of decision-making algorithms was developed: a low-level specific solution-focused set of algorithms, and a high-level hyper-heuristic which selects and seamlessly switches between them, evaluates performance achieved in each step of agent interaction in the monitoring mission, and employs reinforcement learning to enable a level of adaptation to unknown environments, environmental changes, or changes to the agents themselves such as thruster failure. Performance indices were defined to appropriately represent system capabilities and achieved states, primarily related to energy-efficient movement, preserving quality coverage of the monitoring area, and collecting the maximum amount of sensor measurements. Finally, a method of scoring and ranking the performance of the various decision-making methods present in the system was defined, as well as a benchmark to enable validation during a variety of relevant scenarios. These metrics were applied during the analysis of both simulated and field experiment results.

Keywords: hyper-heuristics, heuristics, energy sharing, multi-robot systems, cooperative control, mission planning, autonomous surface vehicles

Hiperheuristički pristup ostvarivanju dugoročne autonomije u heterogenome roju pomorskih robota

U svrhu omogućavanja dugoročne autonomije heterogenog roja morskih robota, u postupke upravljanja energijom sustava i interakcije među agentima uvode se algoritmi raspoređivanja zadataka. U scenariju u kojem višerobotski sustav treba samostalno provoditi dugoročnu nadzornu misiju, raspoloživi maksimalni kapacitet od pet autonomnih površinskih vozila - aPad platformi koje predstavljaju punjače sustava - je obično nadmašen brojem aktivnih zahtjeva za punjenjem, što dovodi do potrebe za pažljivim planiranjem i optimizacijom robotskih aktivnosti. U okviru ovog istraživanja razvija se dvoslojni sustav algoritama za donošenje odluka: na nižoj razini je niz algoritama baziranih na različitim paradigmama strojnog učenja usmjerenih na specifične situacije i rješenja dok je na višoj razini hiperheuristički algoritam koji odabire između njih.

Predloženi predmet istraživanja i primjene je višeslojno društvo podvodnih i pomorskih robota razvijeno kao dio Horizon 2020 FET projekta subCULTron. Cilj ovog istraživanja je postizanje dugoročne autonomije u učećem, samoregulirajućem i samoodrživom roju pomorskih robota koji izvršava nadzornu i istraživačku misiju u izazovnom pomorskom okruženju venecijanske lagune. Heterogeni robotski sustav sastoji se od tri zasebne vrste agenata, od kojih su ovdje dva najznačajnija umjetne školjke (aMussels) koje putuju između morskog dna (gdje djeluju kao senzorske jedinice) i površine, te umjetni lopoči (aPads) na površini vode koji omogućuju razmjenu informacije i energije, a sadrže i mehaničke priključne stanice za punjenje baterija aMussel robota. aPad je preaktuirana površinska platforma opremljena s četiri potisnika, što mu daje značajnu slobodu gibanja, dok aMussel nema aktuatora osim sustava za upravljanje uzgonom. Komunikacija je vrlo važna u distribuiranoj strukturi roja, što znači da se tijekom rada moraju poštovati ograničenja koja se temelje na dometu dvaju dostupnih primarnih načina komunikacije (WiFi i akustički signal). Nadalje, roj mora biti sposoban prilagoditi se promjenama u okolini i stvarnim morskim uvjetima, kako borbom protiv tako i iskorištavanjem utjecaja pojava kao što su morske struje i vjetar.

Naglasak na nenadziranom radu roja i prilagodbi promjenama okoline sugerira primjenu metoda strojnog učenja. Područje umjetne inteligencije može se podijeliti prema mnogim kriterijima, no pet paradigmi strojnog učenja uključuju:

- spojne metode (neuronske mreže)
- genetske algoritme i klasifikatore
- empirijske metode za stvaranje pravila i stabla odlučivanja
- analitičke metode učenja
- pristupe na temelju slučajeva

Predložena struktura sustava za donošenje odluka ima dva glavna sloja: na nižoj razini nalazi se skup algoritama (heuristika) za dodjelu zadataka temeljenih na aspektima nekoliko od gore navedenih paradigmi strojnog učenja, dok algoritam na višoj razini (hiperheuristika) odabire između njih.

Hiperheuristika je općenito definirana kao automatizirana metodologija za odabir ili generiranje heuristike za rješavanje teških računskih problema pretraživanja. Izvorno nazvana "heuristika za odabir heuristika", ona predstavlja pristup na visokoj razini koji može odabrati i na konkretan problem primijeniti odgovarajuću heuristiku na nižoj razini, te to učiniti u svakoj točki odlučivanja. Heurističke i metaheurističke metode uspješne su u rješavanju problema pretraživanja u stvarnom svijetu, međutim nailaze na poteškoće u primjeni na nove probleme ili čak na nove slučajeve vrlo sličnih problema. Glavni uzrok tih poteškoća je širok raspon algoritama i parametara koji su uključeni u rješavanje problema, kao i nedostatak smjernica o tome kako odabrati između njih i kako ih podesiti. Osim toga, razina razumijevanja zašto pojedine heuristike rade u određenim situacijama, a ne u drugima često nije dostatna za jednostavno donošenje izbora, a teškoće u preciznom modeliranju problema i realnih situacija znače da strogo matematički optimalna rješenja možda zapravo i nisu najbolja moguća rješenja u primjeni. Cilj korištenja hiperheuristike jest povećanje razine općenitosti na kojoj optimizacijski sustavi mogu djelovati.

Doktorski rad podijeljen je na uvodni dio ("1. Introduction") u kojem je dan opis glavne motivacije iza teme istraživanja – autonomnog dugoročnog nadzora izazovnih pomorskih ekosustava – i uključuje kratki pregled istraživanja umjetnog života i robotskih društava, s naglaskom na evolucijske algoritme i učenje potkrepljenjem. Poglavlje završava pregledom strukture rada, temeljnih hipoteza i izvornog znanstvenog doprinosa.

Sam robotski roj korišten u radu opisan je u drugom poglavlju ("2. Heterogeneous marine swarm agents and interactions"). Poglavlje sadrži pregled robotskih agenata koji čine heterogeni pomorski roj s naglaskom na agente aPad i aMussel, opisujući hardverska i softverska rješenja koja su razvijena i korištena kako bi se ispunili svi preduvjeti za cilj nenadzirane dugoročne misije praćenja i istraživanja okoliša, uključujući modeliranje agenata koje omogućava testiranje sustava u simulacijama i eksperimentima sa stvarnim vozilima. Poglavlje također opisuje interakcije agenata s ciljem povećanja autonomije roja, posebno naglašavajući kako je postignuta razmjena energije između agenata, zaključno s eksperimentalnom validacijom razvijenih algoritama za autonomno međusobno sakupljanje i punjenje agenata.

Treće poglavlje ("3. Multi-robot task assignment and low-level heuristics") fokusira se na specifični problem raspoređivanja interakcija i zadataka među agentima koji je glavni predmet rada, kao i na razvijenu strukturu sustava za donošenje odluka. Detaljno se opisuju metode podjele i klasteriranja koje se koriste za dodjeljivanje početnih područja

odgovornosti agentima i niža razina sustava donošenja odluka koja se sastoji od situacijskih heuristika koje uključuju znanje o okolini. Poglavlje opisuje i pokazatelje učinkovitosti odabrane za evaluaciju željenih aspekata i mogućnosti sustava. Opisane su i rane simulacije robotskog roja temeljene na diskretnim događajima, kao i početni eksperimenti za validaciju komunikacije i metode postizanja konsenzusa među agentima u roju.

U svrhu osmišljavanja heuristika niže razine, problem dodjeljivanja zadataka aPadima može se opisati kao vrsta problema usmjeravanja vozila. Heurističke metode rješavanja koje su usmjerene na specifične varijante problema usmjeravanja vozila, kao što su problem usmjeravanja vozila s vremenskim prozorima, kapacitirani problem usmjeravanja vozila, problem usmjeravanja vozila s više opskrbnih mjesta ili problem usmjeravanja vozila sa stohastičkim zahtjevima, intenzivno su proučavane, uz razvoj, implementaciju i detaljno prilagođavanje heuristika kako bi odgovarale određenoj vrsti problema. Budući da se obilježja problema mogu znatno razlikovati, nije nužno uvijek sasvim jasno koja će metoda dati najbolje rješenje za određeni slučaj problema. Ovdje je cilj osmisliti nekoliko kontekstualno specifičnih heuristika s dobrim ponašanjem u određenoj situaciji, a zatim odabirati između njih i kombinirati pristupe prema potrebi.

Neki od pristupa uključuju korak odvajanja aMussel robota u grupe ili klastere koji odgovaraju parametrima kao što su ukupna raspoloživa energija, snaga signala ili izvediva udaljenost kretanja, a zatim dodjeljivanje aPadova pojedinačnim klasterima kao oblik određivanja "područja interesa" kako bi se osigurala pokrivenost i u smislu komunikacije i u smislu punjenja. Klasteriranje je problem prisutan u rudarenju podataka, bazama podataka, kompresiji podataka i strojnom učenju. Ono uključuje podjelu skupova opažanja u klastere tako da su intra-klasterska opažanja što je moguće sličnija (ili bliža, u odabranoj metrici), a inter-klasterska opažanja što različitija (ili dalja). Drugi je cilj klasteriranja smanjiti složenost podataka zamjenom skupine opažanja jednim reprezentativnim opažanjem, što dovodi do lakšeg i bržeg računanja i analize.

Algoritmi bazirani na particioniranju, kao što je široko korišteni k-means algoritam, su specifični podtip klasteriranja koji organizira objekte u određeni broj particija, gdje svaka od njih predstavlja jedan klaster. Klasteri se formiraju na temelju funkcije udaljenosti, što dovodi do formiranja samo sfernih klastera i dozvoljava utjecaj šuma na rezultate klasteriranja. Ako se razmatra specifičan problem dodjele zadataka za komunikacijsko pokrivanje i punjenje robota, ovo svojstvo nije osobito problematično, jer su konveksni i sferni klasteri zaista željeni rezultat. K-means algoritam zahtijeva da broj klastera bude unaprijed poznat i zadan, ali se inače može smatrati potpuno nenadziranim ili, u slučaju uključivanja nekih prethodno poznatih saznanja o željenim ishodima, djelomično nenadziranim.

Diferencijalna evolucija je još jedan predstavnik heurističkih pristupa niže razine dod-

jele zadataka i odlučivanja unutar roja. Izvorno je predložena kao iterativna heuristika za globalnu optimizaciju u kontinuiranom prostoru temeljena na populaciji, a kasnije je prilagođena za upotrebu u diskretnim prostorima i u problemima sekvenciranja, permutacije i raspoređivanja, uključujući i problem usmjeravanja vozila. Uvođenjem specifičnog kodiranja gena i populacije i stvaranjem prikladnih funkcija troška, kazni i uvjeta zaustavljanja, moguće je predstaviti i uvesti ograničenja u prostoru rješenja. U ovom će se slučaju kriteriji podudarati s čimbenicima iz stvarnog svijeta kao što su snaga morske struje i razina napona baterije robota, s genima koji kodiraju nizove zadataka za pojedina vozila.

Razvoj prvog sloja sustava za donošenje odluka uključuje razvoj skupa algoritama za dodjelu zadataka koji koriste ili kombiniraju različite paradigme strojnog učenja ili ih koriste na različite načine, s ciljem osmišljavanja rasporeda kretanja i plana izvršavanja zadataka za dostupna vozila, prvenstveno u smislu punjenja ili premještanja drugih robota unutar roja. Ovo uključuje metodu klasteriranja, metodu klasteriranja s ograničenjima, metodu diferencijalne evolucije i hibridne metode. Osim toga, prisutne su vrlo jednostavne i situacijske heuristike razvijene tijekom rada s robotskim sustavom i koje odgovaraju na specifične probleme koji su uočeni. To su Greedy heuristika, koja u svakom koraku odabire najbližu mušulu kako bi se minimiziralo kretanje aPada; Cautious heuristika, koja u svakom koraku bira mušulu s najpraznijom baterijom kojoj je hitno potrebno punjenje; Rescue heuristika, koja bira mušulu najudaljeniju od centroida klastera kako bi se izbjegao gubitak robota i očuvali outlieri roja, te Rush heuristika, koja bira mušulu s najviše baterije, a koja je još uvijek kandidat za punjenje, kako bi se minimiziralo gubljenje korisnog vremena rada mušula. Posebna se pozornost posvećuje rubnim slučajevima u kojima se pojedine metode u određenim uvjetima ponašaju veoma dobro, a u drugima veoma loše (uključujući i uopće ne ostvarivanje konvergencije na valjano rješenje) jer će to biti vrijedan budući pokazatelj uspješnosti za hiperheuristički dio biranja algoritama.

Drugi sloj sadrži hiperheuristički algoritam koji odabire između heuristika implementiranih u donjem sloju ovisno o situaciji u okolini i unutar roja, s krajnjim ciljem autonomnog i nenadziranog planiranja zadataka i istovremenog optimiranja potrošnje i distribucije energije. Pokazatelji učinkovitosti kojima se ocjenjuje rad sustava uključuju broj aktivnih i prisutnih agenata, prostornu raspodjelu agenata unutar roja, te praćenje ukupne količine vremena koje su agenti proveli radeći "koristan" posao, kao i raspodjelu tog vremena.

Viša razina sustava odlučivanja - drugi sloj - opisana je u četvrtom poglavlju ("4. High-level heuristics - hyper-heuristics"). Poglavlje počinje pregledom hiperheuristike u literaturi, uključujući klasifikaciju i općeprihvaćene oblike izvedbe. Zatim se opisuje odabrana i implementirana viša razina strukture odlučivanja, uključujući detaljni opis metode odabira, donošenja odluka i učenja potkrepljenjem koji se koriste kako bi se

sustavu dala prilagodljivost i autonomija. Definirane su metode vrednovanja rješenja koje koriste bodovanje i rangiranje, a u poglavlju su opisane i analizirane i simulirane i eksperimentalne varijante validacijskog referentnog scenarija.

Hiperheuristike se dijele na tri kategorije ovisno o tome uče li tijekom pretraživanja ili prije pretraživanja, ili uopće ne dobivaju povratne informacije iz prostora pretraživanja: hiperheuristika s on-line učenjem, s off-line učenjem i bez učenja. Mogu biti klasificirane i kao selekcijske ili generacijske hiperheuristike, ovisno o tome odabiru li heuristiku iz skupa postojećih heuristika ili generiraju nove heuristike iz komponenti postojećih heuristika niže razine. Hiperheuristika korištena u ovom radu je selekcijska hiperheuristika s online učenjem. Selekcijska hiperheuristika bavi se problemima neizravno, pregledavanjem skupa dostupnih heuristika za svaki korak pretraživanja i odabirom one metode koju će se primijeniti na problem na temelju statistike prethodnih performansi svih metoda prema danom skupu metrika i pokazatelja.

Strategija odabira ruletom često se koristi u evolucijskim algoritmima. Primijenjeno na hiperheuristički okvir, rulet odabire heuristiku s vjerojatnošću proporcionalnom njezinoj sposobnosti. Modificiranjem parametara sposobnosti metoda na temelju povratnih informacija primljenih iz prostora problema korištenjem više pokazatelja učinkovitosti, online učenje s potkrepljenjem uvodi se u sustav donošenja odluka. Nakon svake primjene heuristike niže razine, ona se ocjenjuje na temelju odabranih pokazatelja. Ako je heuristika imala bolju učinkovitost od dotadašnjeg prosjeka s obzirom na određeni pokazatelj, ona se pozitivno ocjenjuje za taj pokazatelj i njezina se sposobnost povećava za svaki takav "uspjeh". U suprotnom, ako heuristika ima lošiju učinkovitost, smatra se da nije uspjela i dobiva fiksnu "kaznu" koja umanjuje njezinu sposobnost i time joj smanjuje vjerojatnost ponovnog odabira.

Hiperheuristička struktura znači rad odvojen od problemske domene, sa svim aspektima stvarnog svijeta apstrahiranim u pokazatelje koji se nakon implementacije i evaluacije vraćaju hiperheuristici. Ako, na primjer, tijekom eksperimenta struja, vjetar ili kvar na vozilu otežaju aPadu da se pomakne i dosegne aMussele odabrane u predloženom rješenju, trošak kretanja bit će znatno veći nego ranije u eksperimentu, a heuristika niže razine koja je proizvela ovo rješenje bit će kažnjena, dok će ona metoda koja predloži rješenje koje može na bilo koji način zaobići ili kompenzirati ovu smetnju i smanjiti troškove kretanja biti nagrađena, čime se postiže postupna adaptacija čitavog sustava.

Zbog prirode sustava na koji se primjenjuje, učenje s potkrepljenjem mora biti u stanju donositi zaključke i odgovarajuće prilagodbe sposobnosti na temelju relativno oskudnih i rijetkih povratnih informacija. Budući da optimalno rješenje ovdje nije poznato i računalno ga je previše zahtjevno pronaći, kako bi se ocijenila izvedba implementirane hiperheuristike, usporedbe se rade pomoću najboljeg postignutog rješenja. Hiperheuris-

tički odabir s učenjem s potkrepljenjem trebao bi biti bolji od jednostavnog odabira jedne od heuristika i njezine kontinuirane upotrebe ili nasumične primjene dostupnih heuristika.

Učinkovitost se procjenjuje na temelju četiri odabrana pokazatelja: vrijeme rada aMussel robota, troškovi kretanja aPada, očuvanje outliera među aMussel robotima i (ne)uravnoteženost distribucije rada aMussela. U slučaju neravnoteže troškova kretanja i vremena rada, niža ocjena je bolja (što implicira da su kretanje i potrošnja energije aPada minimizirani, a koristan rad i vrijeme rada bili su ravnomjernije raspoređeni među svim agentima), dok je viša ocjena bolja za očuvanje outliera i postignuto ukupno vrijeme rada aMussela (što znači da su mjerenja konzistentno dobivana i od udaljenih agenata).

Različiti eksperimenti ocjenjuju se korištenjem metoda rangiranja. Metoda koja pronalazi najbolje rješenje u skupu eksperimenata koji se ocjenjuju dobiva najnižu vrijednost, dok metoda s najlošijom izvedbom dobiva najveću vrijednost, a sve ostale se nalaze između. Metoda s ukupnim najnižim rangom stoga se može smatrati metodom s najboljim ostvarenim uspjehom. Tako su predstavljena i vrednovana četiri simulirana eksperimenta/scenarija: osnovna simulacija roja bez smetnji ili ekstrema, dvije instance u kojima se smetnja javlja u tijeku eksperimenta što dovodi do promjene u sposobnostima agenata, a time i ponašanja cjelokupnog roja, i jedna instanca u kojoj postoji jedan aMussel agent postavljen na znatnoj udaljenosti od ostatka roja. Eksperimentalni scenarij ekvivalentan osnovnoj simulaciji bez smetnji izveden je i sa stvarnim aPad vozilom.

Rezultati simulacija i eksperimenata potvrđuju hipotezu da je moguće kontinuirano generirati nizove jednostavnih i računski nezahtjevnih heuristika koje su po definiranim pokazateljima učinkovitosti uspješnije od opetovane primjene svake pojedinačne heuristike, kao i njihove nasumične primjene. Kao hiperheuristička struktura s čvrsto postavljenom domenskom barijerom, sustav može generalizirati na različite nove situacije koje se pojavljuju u problemskom prostoru, npr. promjene u vrstama, broju i mogućnostima agenata roja, promjene u uvjetima u okolini roja i rad u različitim okolinama općenito.

Temeljem upravljačkih algoritama i metodologija za validaciju algoritama razvijenih unutar doktorata izdvojena su tri znanstvena doprinosa:

- Metoda dodjele i nizanja zadataka za više robota koji osiguravaju dugoročnu autonomiju heterogenog roja pomorskih robota uzimajući u obzir ograničenja okoline.
- Hiperheuristička metoda za donošenje odluka unutar heterogenog roja pomorskih robota temeljena na nenadziranom odabiru između metoda dodjele i nizanja zadataka.
- Metoda vrednovanja rješenja i definicija pokazatelja učinkovitosti i referentnog scenarija za procjenu valjanosti metoda donošenja odluka i dodjele zadataka primijenjenih na heterogenom roju pomorskih robota.

Doktorski rad završava pregledom hipoteza i gore navedenih doprinosa te sažetkom

najvažnijih točaka disertacije. Na temelju prezentiranog sadržaja ponovno se postavljaju te detaljnije razrađuju hipoteze i kao dokaz inovativnosti istraživanja nudi se popis publiciranih znanstvenih radova.

Ključne riječi: hiperheuristika, heuristika, dijeljenje energije, višerobotski sustavi, kooperativno upravljanje, planiranje misije, autonomna površinska vozila

Contents

Chapter 1

Introduction

The Lagoon of Venice, Italy is a very particular ecosystem, highly relevant in a scientific, environmental, cultural, and socio-economic context and a critical subject of study. It is also an area undergoing significant shifts, severely impacted by both global climate change-related effects as well as site-specific phenomena [1] [2]. With such environmental factors as abundant spring rains and intense summer heatwaves, along with eutrophication and the effect of intense human activity on various types of sediment and soils present in the lagoon, notable phenomena include degradation of zooplankton, abnormal proliferation of certain species of macroalgae such as *Ulva rigida*, and hypoxic and anoxic events in the form of rapid localised (usually overnight) drops in the concentration of oxygen in the water. Hypoxic and anoxic events are becoming more frequent and more spatially widespread, and have been tied to fish mortality and biomass shifts [3] [4] [5] [6].

The Horizon 2020 Future and Emerging Technologies project subCULTron (submarine CULTures perform long-term robotic exploration of unconventional environmental niches) [7] was conceptualised as a novel answer to the demand for long-term environmental monitoring in the waters of Venice. The goal of the project was superseding the need for boats and personnel periodically deploying and collecting probes at sea and at a variety of inaccessible locations (such as extreme shallows, marshes, areas with traffic-related limitations, or remote areas) by developing an autonomous heterogeneous swarm of marine robots, shown in Figure 1.1. In effect, creating a topologically reconfigurable underwater acoustic sensor network with surface access points [8], as well as an unsupervised multi-layer learning, self-regulating, self-sustaining underwater robot society. The main motivation of this thesis is achieving long-term sustainability and survival of this robotic swarm, both to ensure environmental monitoring mission success, and to enable the study of emergent phenomena within this artificial ecosystem, or robotic behaviours on a "societal" scale.

Starting in the mid-1980s, the scientific approach to the field of artificial life has been

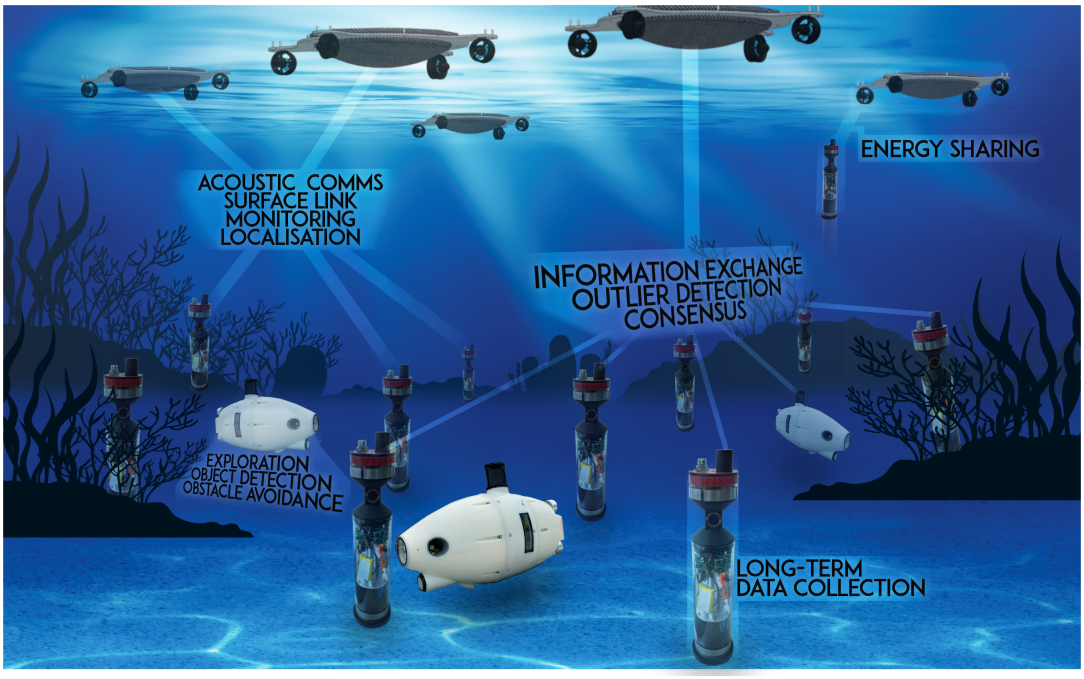


Figure 1.1: subCULTron heterogeneous marine robot swarm concept illustration highlighting interactions between different agent types.

to study living systems using a synthetic "bridge": building life, as hardware or software or even wetware, in order to reach a better understanding of it. While intensely interdisciplinary, artificial life research can be classified into fourteen main themes: origins of life, autonomy, self-organization, adaptation (including evolution, development, and learning), ecology, artificial societies, behaviour, computational biology, artificial chemistries, information, living technology, art, and philosophy [9].

Reinforcement learning is a common form of machine learning inspired by behaviourist psychology, and includes adaptation occurring through agents interacting with their environment. Standing apart from the classic paradigm of learning from a set or sequence of examples, reinforcement learning instead opts for a trial and error approach, featuring a reward provided by an interpreter evaluating these agent-environment, and potentially, in multi-agent systems, agent-agent interactions [10].

Early steps towards an increasingly behavioural approach to robotics and viewing behaviour as emerging from the robot-environment interaction, while relying on a grade of autonomy that makes micromanaging all aspects of a system unnecessary (and undesirable), were made in [11]. Explicitly encouraging behavioural diversity using behavioural distance/diversity mechanisms has been empirically shown to lead to substantial performance improvements in several typical evolutionary robotics experiments [12]. The evolution of cooperation, frequently based on game theory, is an increasingly popular research topic [13].

Several novel frameworks for studying cognitive systems have arisen from devising and implementing distributed approaches to the problem. Swarm cognition is one such framework, and it promotes the study of cognition as an emergent collective phenomenon, and as something that can be recognised in the behaviour of entire collectives, including complex societies [14].

A useful way of classifying robot societies is according to modes of interaction between their members. The most common and relevant forms these interactions take are:

- collective, where the robot entities are not aware of each other, although they share goals and their actions benefit each other and the team as a whole. An example of this type of interaction in multi-robot systems is a robotic swarm. An individual robot in this kind of society likely has a very simple controller and structure, but by interacting with others a global goal (some sort of flocking, formation-keeping, or foraging and similar) is achieved, often as an emergent property of these many smaller local interactions.
- cooperative, where all members of a team of robots that is trying to accomplish a common goal are aware of each other, and their actions are mutually beneficial. Robots in a society that functions this way may work together and reason about each other's capability to contribute to the completion of a given task. At times, the individual robots may be working on different parts of the higher goal, and so need to ensure that they don't interfere with each other's work. Optimal task allocation and coordination are very important, even while the majority of the robots' operation is focused on working together towards the common goal.
- collaborative, where robots have individual goals, but are aware of the other robots on their team, and their actions advance both their own and others' goals. Each member of a group of robots of this type has its own agenda, all agendas are compatible with each other, and the robotic agents might be heterogeneous with regards to sensor/actuator setups and related properties. In this kind of society, focus is on how the unique expertise of the individual can contribute to another individual achieving its goal and thus the ultimate goal of the team as a whole, by bringing together agents with complementary skills and assets. Coalition formation is a notable type of collaboration.
- coordinative, where all robotic entities are aware of each other, but do not share a common goal, and their actions are not helpful to the rest of the team. Typically this implies several robots sharing a common workspace, and thus needing to coordinate among themselves to avoid interference and mitigate any potential negative outcomes of intra-robot interaction. Multi-robot path planning techniques and collision avoidance are a good example of this.

Additionally, the interaction between robots can be adversarial, i.e. the goals of some agents may have a negative effect on others. Two rival teams of robots attempting to accomplish the same task while hindering each other's progress would fall under this category, a popular example of which is robotic soccer [15].

The key to successfully performing advanced tasks in complex modern-day environments is the emergence of cooperative abilities among robots forming a society. Two types of robot societies can be considered when evaluating group cooperation ability: integrating and differentiating. Integrating robot societies consist of a smaller number of heterogeneous individuals with highly specialised skills. Examples of biological equivalents include wolf packs and bird colonies. Differentiating robot societies, on the other hand, consist of a large number of homogenous individuals with relatively limited abilities. A biological equivalent to this type of society would be insect colonies such as those of bees and ants. Notably, both types of societies require individuals to have well-defined roles and the ability to dynamically modify their behaviour while the group is performing an assigned task [16].

While the field of evolutionary robotics typically studies the use of evolutionary algorithms in an offline fashion - i.e. running a set of algorithms and optimising the robots' controllers in simulation software before any transfer to actual robotic hardware, use, and deployment, and without any additional adaptation as the robots operate - the concept of embodied evolution, espoused by the embodied evolutionary robotics subfield, focuses on implementing evolutionary algorithms on the hardware itself and having them run during the robots' operational period - an online approach. A marked advantage to this is giving the robotic system a chance to acquire beneficial behaviours as it functions, without a human in the loop, thus giving it an ability to adapt to previously unknown environments or environmental conditions that change through time [17].

The field of robotics has been the most active in the development of embodied evolution algorithms mainly due to the fact that the intrinsically adaptive and self-organising properties of the algorithms make them highly appropriate for use in real-time autonomous systems that a group of robots presents. There are two main approaches that have arisen in the study of embodied evolution: encapsulated embodied evolution algorithms, where each individual agent carries an entire population of controllers upon which an independent evolutionary algorithm is run; and distributed embodied evolution algorithms, where each individual agent in the population carries only its own genotype, necessitating extensive interaction between agents.

Distributed embodied evolution algorithms offer a much bigger potential when it comes to the emergence of a variety of self-adaptive and cooperative behaviours, due to the complex interactions that can occur within the population. Their drawback is that they

converge less easily and require larger populations of robots to do so at all. They are also very sensitive to their configuration parameters, requiring close task-dependent regulation to ensure that a valid solution is reached [18]. Embodied evolution can also be described as taking place in a population of robots where selection, evaluation, and reproduction, the main mechanisms of evolutionary algorithms, are carried out by and between the robots, all in a distributed, asynchronous, and autonomous manner [19].

Thus, this approach seems naturally suited for work with multi-agent systems attempting to solve complex but structured problems via decentralised collaboration, as there exists a need in multiagent systems to coordinate local policies of each agent with their restricted capabilities to achieve a system-wide goal. The presence of innate uncertainty in this type of system adds to the complexity of this task, as agents need to learn unknown environment parameters while forming these local policies in an online fashion [20].

Although the agents in a system can be pre-programmed with a set of behaviours designed in advance, the learning of new behaviours online is often necessary to ensure the gradual improvement of the entire system's performance [21]. Once again, the main issues that arise when considering a more offline approach are a dynamically changing, complex, and not entirely known environment, making it a very real possibility that a hardwired behaviour may at one point become inappropriate or even outright negatively affect performance. The benefits of multiagent reinforcement learning arise primarily from the distributed nature of the multiagent system, and include a certain computational speedup made possible by parallel computation, and positive effects of sharing experience between multiple agents taking the form of communication, teaching, and imitation.

Beyond challenges that are inherited from classical single-agent reinforcement learning, such as the problem of dimensionality and the need to strike a balance between exploration and exploitation (or task completion) by careful reward selection, a multiagent approach also engenders difficulties in the form of a need for coordination and the challenge of specifying a learning goal. The presence of multiple agents in the same environment, learning in parallel, complicates matters considerably, as the action space scales exponentially with the number of agents. A notable consequence of this is that some standard learning techniques that store a reward value for every possible state-action combination become unfeasible. Another issue is that the behaviour of one agent influences the outcomes of other agents' individually selected actions, thus incurring change in the environment and possibly compromising convergence [22][23].

The design of the reward function is a critical part of achieving a real-world task by using reinforcement learning and, although it may appear straightforward, it can demand careful tuning when multiple rewards are present. An evolutionary approach to mul-

tiagent reinforcement learning includes the so-called intrinsically motivated framework, based upon several psychological theories of motivation, in which agents themselves find appropriate intrinsic rewards that implicitly help task success or affect the fitness of the agent (while striking a balance between rewarding exploration and task completion, the previously mentioned common problem in reinforcement learning design). Evolutionary algorithms also optimise meta-properties in reinforcement learning by influencing the selection of behaviours, modulating the efficiency of learning by affecting learning meta-parameters, or changing an forming entirely new reward signals that guide the reinforcement learning process [24][25][26][27].

Designing an online evolutionary algorithm distributed among a fixed population of autonomous robots for the purposes of long-term survival and operation in a given environment greatly depends on the chosen fitness function for the environment-driven distributed evolutionary adaptation. The implicit nature of this fitness function is the consequence of two complementary or conflicting motivations that are present in agents:

- extrinsic motivation, which arises purely from the interaction between an agent and its environment (with or without other agents) and states that an agent must maximise its chances of survival while working within environmental constraints and overcoming environmental challenges
- intrinsic motivation, which arises from the agent itself, and which states that, in keeping with the evolutionary process, a certain set of parameters (or genomes, as they are understood here) must spread across the agent population in order to survive, meaning the genomes promote maximising the number of agents met and interacted with (thus maximising their own opportunities for proliferation).

In a situation where the motivations are well-balanced and the environment-driven distributed evolutionary adaptation algorithm is an efficient one, a state of equilibrium should be achieved in which the genome that can be considered optimal reaches maximum spread while preserving maximum possible survival efficiency. While there would appear to be inherent conflict between these two motivations, it is possible for them to correlate enough so that they can be treated as one motivation, greatly reducing the complexity of the problem. The application of these motivations and a focus on the effect of the environment on the evolutionary adaptation process in embodied evolution is an important concern, since embodied evolution concerns itself primarily with a fixed number of physical agents meant to operate in real-world environments - meaning their most basic operation can be severely impacted by obstacles or energy constraints [28].

An important thing to note is that the study of the development of robot controllers or robot morphologies via the use of artificial evolution does imply a certain need to rely on simulation environments during development, since the duration of a single experiment

can be impractically long, and the necessity of trial-and-error interactions with the environment during attempts at reinforcement learning can lead to damage of actual physical robotic platforms [29].

However, caution needs to be exercised while working within simulated environments, as the generation of simulators introduces a strong bias into the system by way of using symbols and models to represent real phenomena the way they are perceived by the person developing the simulation platform. Additionally, simulating the interaction between the robots and their environment can be very computationally demanding, especially in the case of multiagent systems, thus necessitating considerable simplification to start with. This clashes with some of the fundamental attributes of the evolutionary approach, specifically that it encourages the autonomous generation of robotic systems while leaving out a considerable amount of designer bias, and presents, theoretically, an open-ended approach, while allowing the generation of controllers that are optimally suited for a given operational environment. In any case, the reality gap, or the substantial difference between simulation and reality, persists, and needs to be taken into account when considering robotic behaviour generation.

Closing the reality gap would mean that there exists no difference between simulation and reality, from the viewpoint both of the internal agent - the robot controller - and the external observer - the experimenter evaluating the generated behaviour. This is not possible to completely achieve when doing real-world experiments, so care must be taken to ensure algorithms and desired behaviours are robust with regards to the reality gap, for example by modifying experimental setups to compensate for the reduction in stability that occurs with a decrease in population (as a population of physical robots will almost always be smaller than a simulated one) [30].

Since it is hard to arrive at a satisfactorily encompassing definition, the term emergent is frequently described as something that is "more than the sum of its parts". In a robotic context, emergence is taken to mean that a robot's behaviour has become something not explicitly defined in its controllers, but something that has arisen as a consequence of its interaction with its environment.

The term reactive control refers to the coupling of perception and action in a way that ensures robots respond in a timely manner to moving and working in dynamic, unstructured, and at least partially unknown environments. It also represents a behavioural approach to robotic applications, which is more suited to the complex online context than the classical hierarchical and decision-based approach [31].

A group of robots can be seen as embodying the "more than the sum of its parts" concept when it becomes a robotic team: once it shows some degree of specialised aptitude of performing a task cooperatively, in the sense that the group provides better performance

Figure 1.2: Agents of the subCULTron artificial ecosystem on display in Venice.

than its individual components would, by taking advantage of its distributed sensing and acting capability to carry out complex tasks while also taking into consideration increased fault tolerance thanks to agent redundancy and group cohesion obtained from formation-keeping algorithms and related trajectory calculations and motion planning. Another emergent phenomenon is collective intelligence, the result of two or more agents engaged in global behaviours - meaning an intelligent multi-robot system arises from a group of mobile robots that cooperate, communicate, and dynamically reconfigure their group during their attempts to solve a complex task [32][16].

The concepts of embodied evolution, multiagent reinforcement learning, and distributed control interact and even overlap in a myriad of interesting ways. Furthermore, an empirical approach using the tenets of embodied evolution is considered highly promising, as testing on actual physical systems provides valuable insight into the simulation-reality differences present in a robot society, as well as an opportunity to study novel behaviours that might potentially emerge from a combination of agent-agent and agent-environment interactions.

The heterogeneous robotic system which is the main subject of study within this thesis consists of three separate agent types, of which two are particularly significant: artificial mussels (aMussels, 120 in swarm) which travel between the seafloor (where they act as sensor hubs) and the surface, and artificial lily pads (aPads, 5 in swarm) on the surface of the water, enabling an exchange of information, while also providing mechanical docking stations for aMussels to attach to in order to recharge their batteries or be transported to another location. The different types of robotic agents are shown in Figure 1.2.

Communication is very important in the distributed structure of a swarm, meaning that constraints based on communication range for the two primary modes of communication employed (Wi-Fi and acoustic) need to be taken into account during operation. Additionally, the swarm needs to be able to adapt to a changing realistic marine environment (Figure 1.3), both combating and taking advantage of the influence of phenomena such as water currents and wind, or dealing with ongoing disturbances such as thruster

Figure 1.3: subCULTron swarm agents in outdoor testbed pool (left) and in realistic conditions in the Venice Arsenale (right).

failure. One of the energy-efficient aPad behaviours being explored in the system, adapting to changes in their environment even when in a mostly idle state (i.e. not actively performing tasks), is outlined in ([33]).

In a scenario where the system needs to autonomously go about its monitoring mission and survive long-term, the available maximum of five aPads is usually outnumbered by the number of active aMussels and their charging or transportation requests, leading to a need for careful planning and optimisation of their activities. For the purpose of enabling long-term persistence of the subCULTron system, the concept of aPad task allocation and sequencing was introduced into the energy management procedures of the swarm.

The emphasis on unsupervised functioning of the robotic swarm and adapting swarm behaviours to changing environmental factors and situations suggests the application of machine learning. While the field of artificial intelligence can be divided according to many criteria, the five paradigms for machine learning, as given in [34], include:

1.connectionist methods (neural networks)

2.genetic algorithms and classifier systems

3.empirical methods for inducing rules and decision trees

4.analytic learning methods

5.case-based approaches

The proposed decision-making structure for the swarm has two main layers: a lower-level collection of task allocation and sequencing algorithms based on aspects of several of the aforementioned machine learning paradigms, and a higher-level algorithm that selects between them, evaluating the state of the swarm at every step and adapting and learning all the while - a hyper-heuristic. Performance indices need to be carefully selected in order to abstract swarm agent behaviour into simple numerical scores for the hyper-heuristic to handle, evaluate, reward, and base decisions upon. Finally, defining representative benchmark scenarios to test the swarm's decision-making capabilities is crucial, both in a simulated environment and in real-world conditions.

1.1 Thesis Contribution and Overview

The objective of the thesis is to establish long-term autonomous, unsupervised, energy-efficient, and environmentally adaptive collective decision-making of a heterogeneous swarm of marine robots. The stated hypotheses are as follows:

- *Unsupervised decision-making algorithms with incorporated knowledge of the environment can significantly extend autonomous functioning of a heterogeneous marine robot swarm.*
- *Possible negative impacts of environmental factors such as water currents and wind, as well as vehicle faults and disturbances, can be minimised and adapted to during continuous operation of a robotic swarm.*
- *Interaction within a heterogeneous marine robot swarm can lead to enhanced exploration and monitoring capabilities as well as extended autonomy.*
- *It is possible to devise a benchmark scenario and quantitative metrics to compare various (hyper-)heuristic decision-making algorithms.*

In the scope of this thesis, a two-layered system of decision-making algorithms was developed: a low-level specific solution-focused set of algorithms, and a high-level hyper-heuristic which selects and seamlessly switches between them, evaluates performance achieved in each step of agent interaction in the monitoring mission, and employs reinforcement learning to enable a level of adaptation to unknown environments, environmental changes, or changes to the agents themselves such as thruster failure. Performance indices were defined to appropriately represent system capabilities and achieved states, primarily related to energy-efficient movement, preserving quality coverage of the monitoring area, and collecting the maximum amount of sensor measurements. Finally, a method of scoring and ranking the performance of the various decision-making methods present in the system was defined, as well as a benchmark to enable validation during a variety of relevant scenarios. These metrics were applied during the analysis of both simulated and field experiment results.

Therefore, the scientific contribution of this thesis is summarized as follows:

1. **A method for multi-robot task assignment and sequencing ensuring long-term autonomy of a heterogeneous swarm of marine robots while taking into account environmental constraints.**
2. **A hyper-heuristic decision-making method for a heterogeneous swarm of marine robots based on unsupervised switching between multiple task assignment and sequencing methods.**
3. **A solution evaluation method and definition of performance indices and a benchmark validation scenario for decision-making and task assignment methods implemented on a heterogeneous swarm of marine robots.**

The thesis is organized as follows. In Chapter 2, an overview of the robotic agents within the heterogeneous marine swarm is given, including hardware and software solutions that were employed in order to meet all the prerequisites for the goal of an unsupervised long-term monitoring mission, including agent interactions aimed at extending the swarm's autonomy. Chapter 3 outlines the problem being studied, as well as the structure of the decision-making system being developed. It details the lower level of the system, including the situational heuristics which incorporate knowledge of the environment, and presents performance indices chosen to evaluate desired aspects and capabilities of the system. Chapter 4 describes the higher level of the decision-making structure, including a full examination of the selection method and reinforcement learning employed to give the system adaptability and autonomy. Benchmark methods using scoring and ranking are defined, and simulated and experimental variants of the validation scenario are described. The thesis is concluded with Chapter 5 where a summary of the most important points from the dissertation is given.

Chapter 2

Heterogeneous marine swarm agents and interactions

2.1 Introduction

This thesis includes investigation and validation of various algorithms applied to a heterogeneous swarm of marine robots, and includes implementation as one of its contributions. A key requirement is thus a suitable fleet of marine vehicles.

The subCULTron multi-agent system was envisioned as an artificial marine ecosystem consisting of three agent types. Within the scope of this thesis, the focus is on the use cases, interactions, and interfaces of the aPads and aMussels (Figure 2.1). A detailed description of the functionalities and agent interactions featuring the aMussel and aPad robots, as well as the subCULTron system as a whole, is given in [35]. An overview is provided here for context. An example of a fully implemented monitoring mission scenario for the swarm, demonstrating a variety of interactions between same and different agent types, is described in [36].

The subCULTron swarm can in many ways be characterised as an Underwater Acoustic Sensor Network (UASN). In recent decades, deployment and use of USANs has been growing in popularity [37, 38, 39, 40, 41, 42]. Where traditional underwater monitoring systems utilise expensive and complex individual agents and subsystems for data collection, UASNs replace these individual monitoring systems with smaller and less expensive underwater sensor nodes housing a wide variety of sensors - temperature, pressure, turbidity, and salinity sensors, among others. Additionally, these underwater nodes use acoustic methods of communication and localisation.

In [43] the authors demonstrated an underwater swarm system consisting of heterogeneous robots used for ocean exploration. Underwater sensor nodes use buoyancy control for depth control, while underwater localisation is aided by acoustic-capable surface buoys.

While the swarm described in [43] passively explores the environment relying exclusively on ocean currents, the approach to the subCULTron swarm places greater focus on the ability to dynamically relocate the swarm for planned exploration. Furthermore, the subCULTron system is capable of collective decision making enabled by underwater communication. Due to energy sharing between surface and underwater robots, deployment can be prolonged.

In project Argo [44] a system for ocean profiling was developed, with underwater robots capable of depth control using a variable buoyancy system. The robots have the capability of acquiring underwater measurements while drifting in the ocean, while final data transfer to a remote data hub is done through satellite communication, allowing wide deployment coverage. Each robot is a standalone unit with no planned interaction between the agents or demonstrable swarm behaviours.

Among the several possible interactions between robotic agents in the subCULTron swarm, hardware and software has been developed to enable agents to conserve and share energy [45]. Actively prolonging the operational time of the swarm is done by having an over-actuated autonomous surface platform dock up to four floating sensor nodes at a time and replenish their batteries using wireless inductive charging.

Energy management in robotic swarms is generally addressed by implementing energy-efficient behaviours and providing some method of recharging to prolong swarm autonomy while not adversely impacting the mission in progress [46]. In [47] and [48] the authors aim to achieve arbitrary operating times and continuous energy (re)supply by modifying the operating environment of the mobile ground robot swarms. More applicable to the subCULTron swarm, several approaches have been proposed which focus on energy exchange between members of the swarm, with specialised agents serving as the starting points of energy supply chains [49] or navigating within the swarm while offering use of their multiple charging stations, monitoring other agents' location and battery levels, and responding to charging requests [50].

A self-adaptive algorithm developed to ensure that important swarm properties, such as formation and provided coverage, are preserved during energy-aware behaviours of a group of robots dealing with a limited power source and seeking to continuously recharge is presented in [46]. The robots use only locally available information such as range and bearing with regards to other nearby robots, and very limited communication, making the algorithm suitable for consideration in challenging environments.

Authors in [49] address the potential bottleneck of charging stations serving multiple robots by introducing collaborative behaviours focused on energy sharing between the swarm agents. Some of the robots in rely on charging stations for energy resupply, while others receive energy from peers in their proximity, leading to efficient energy supply

chains. The robots do not spend time recharging, but physically exchange battery modules, and a controller making rendezvous behaviour required to achieve this possible is developed.

As opposed to more traditional docking/charging stations, [50] proposes an autonomous mobile charger robot which acts within the swarm, monitors the other agents' location and battery level, and seeks out low-powered agents to respond to their charging requests. Robots in need of charging approach the charging robot and dock to one of its several charging stations. It also contains algorithms developed for cooperation between robots in need of charging and the robot carrying the charging stations.

Working in the marine environment poses unique challenges. In [51], the authors aim to achieve an energy-efficient underwater swarm by having swarm members adjust their behaviour depending on their energy levels and needs. Similarly to the subCULTron swarm, acoustic communication is used for relaying information from agents on the surface to those that are submerged, as well as for underwater localisation. In turn, [52] proposes an underwater wireless charging system consisting of a station which vehicles navigate to in order to recharge their batteries.

Figure 2.1: The aPad platform and four aMussel underwater sensor nodes.

Due to the logistically demanding nature of experiments in the field, especially in marine environments and with a great number of robots, simulated environments and agents have been created which enable initial testing and validation. Evaluating the performance of different approaches and tuning the many parameters present in the system to improve its longevity is a challenging task and requires being able to repeat experiments several times. Hence, a desire to create an experimental framework in which simulated

agents interact with real ones, in the spirit of the Hardware–In–the–Loop (HIL) approach (or in this case vessel or Vehicle–In–the–Loop (VIL) [53] [54]). While there have been some attempts at standardisation, existing simulators focusing on underwater environments and aspects relevant to marine vehicle development are not widely available and have some VIL capabilities [55] [56], with the more common approach being partial HIL, such as featuring the main electronics boards of the vehicle and one or more real sensors, with the rest simulated [57]. The simulator that is part of the system proposed in this thesis does not feature a dedicated visualisation component, instead being compatible with existing command and control frameworks. It also aims to simulate underwater agents in their entirety, with seamless switching between simulated and real agents.

The simulations were made in Python using the SciPy library and the functionalities of the Robot Operating System (ROS) distributed runtime environment [58], widely used in the robotics community. The method of implementation used allows for a very simple transfer onto real hardware and initial validation by hardware-in-the-loop simulations. It also makes it possible to mix real and simulated vehicles during experiments – for example, a single aPad is relatively simple to deploy, compared to 24 aMussels: therefore, experiments with actual aPad vehicles "charging" many simulated aMussel agents are very useful for algorithm validation. Simulated aMussels can be run on a dedicated surface station computer, or can be run on the on-board computer of a "host" aPad. Low-level control and navigation of the aPad vehicles is readily available, as are their kinematic and dynamic models, whereas the aMussel agents required additional modelling, in particular with regards to battery charging and discharging.

This chapter gives an overview of the main two agents studied in this thesis from both a hardware and software standpoint as well as their representation in developed simulated and VIL environments, with a particular emphasis on subsystems enabling the newly developed agent interaction of autonomous docking and charging, a prerequisite for all algorithms described in this thesis.

2.2 The aPad robotic agent type

The aPad is a highly manoeuvrable modular robotic platform with powerful batteries and the highest computational power in the swarm. Equipped with mechanical docking stations and inductive charging coils, one of the aPad's main roles and most important abilities is to transport and wirelessly charge up to four other swarm agents at a time (Figure 2.2). The aPads each act as an energy and information sharing hub, as well as a bridge between the swarm and a potential human observer. They also serve as anchors in the process of acoustic underwater localisation of other agents.

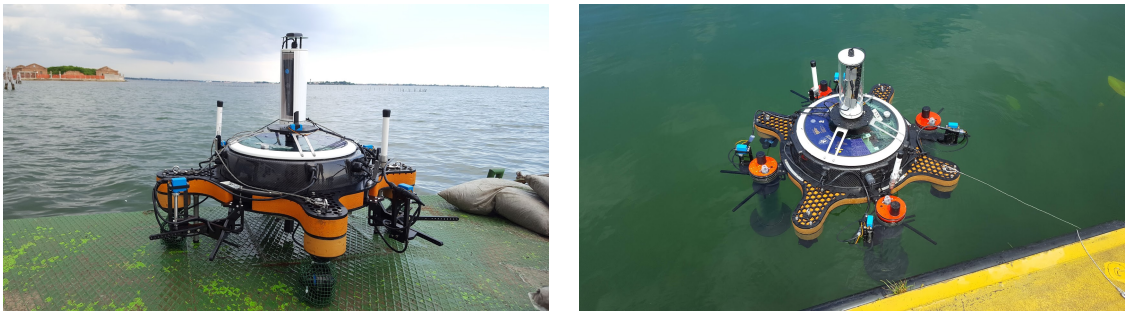


Figure 2.2: Final version of the aPad platform ready for deployment on-site (left) and an aPad fully loaded with four docked aMussels (right).

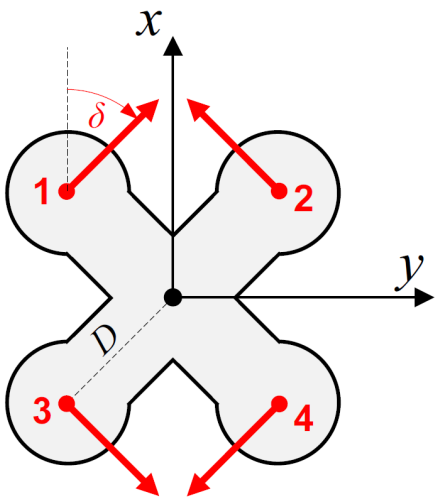


Figure 2.3: X-shaped thruster configuration present on the aPad, enabling omnidirectional motion in the horizontal plane, [59].

The vehicle is 0.385 m high, 0.756 m wide and long, and weighs approximately 25 kg. It has four thrusters in a specific X-shaped configuration (Figure 2.3) which make it over-actuated and omnidirectional. It can return to a home position and be charged while deployed, without the need for recovery and redeployment, thanks to a waterproof charging jack on its hull. For surface communication purposes, the aPads are equipped with a mesh-capable wireless router which operates on two different frequency bands using a separate antenna for each, providing WiFi access points for aMussels and surface stations and a mesh network for the aPads themselves. For underwater communication and localisation purposes, the aPads use miniaturised acoustic modems called nanomodems, while for localisation and positioning, each platform has an Inertial Measurement Unit (IMU) and a Global Positioning System (GPS) module with the capability of using the Real-Time Kinematic (RTK) positioning technique for greater precision.

In addition to the exchange of information, the aPads also have the ability to exchange energy with other types of agents via the use of specially designed mechanical docking stations and an autonomous docking algorithm. The aPad seeks the recognisable top cap

of a floating aMussel using a visual sensor, approaches it using a control algorithm based around visual servoing, moving so the aMussel docking section is aligned with the aPad docking mechanism, then finally activates the docking mechanism to secure the aMussel in place so wireless charging using inductive coils can take place, or so the aMussel can be transported to another location.

The aPad visual sensor is a Microsoft Kinect Infra-Red (IR) and Red-Green-Blue (RGB) combination sensor encased in a watertight plexiglass tube, mounted on a specially designed motorised pan mechanism which allows it to turn and look down over each of the four docking mechanisms (Figure 2.4). A sun protection sticker was applied to the plexiglass tube in order to prevent the sensor overheating during prolonged deployment and operation in direct sunlight. The platform's GPS antenna is additionally mounted on top of the tube in order to ensure the best signal reception possible.

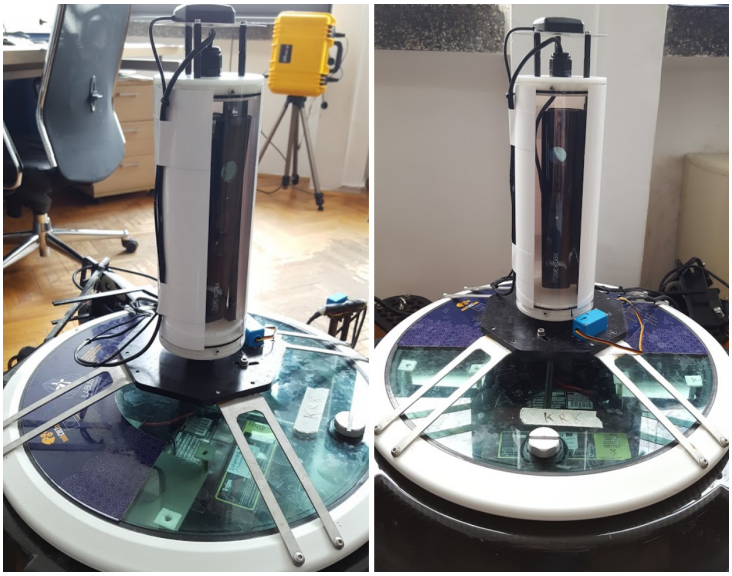


Figure 2.4: Final setup of the Microsoft Kinect sensor sideways within a watertight tube, mounted on top of the aPad platform with a motorised pan mechanism.

The charging dock on the aPad contains three charging coils encapsulated in water-proof resin which provide wireless energy transfer, and is shaped in a way to provide tolerance in aMussel angle at approach, as well as mechanical locking once the aMussel is in the slot. Two immovable delrin levers are mounted on the aPad which act as a guiding rail funnelling the aMussel towards the charging dock, while a motorized shutter pushes on the aMussel while closing. The docking principle is purely mechanical, designed to have quite high tolerances to offsets with the approaching aMussel, including vertical shift tolerance of $\pm 50mm$, and horizontal shift tolerance of $\pm 130mm$.

After several field and stress tests, the docking mechanism was modified to keep the servo motor moving the arm well above water (as its water resistance rating proved not

high enough to tolerate repeated complete submersion). This also reduced the force it needed to exert during closing by ensuring better leverage, thus avoiding excess mechanical wear and increasing the longevity of the motor as well as the entire mechanical system. The design of the dock itself was modified and streamlined by thinning and lengthening its two prongs for more horizontal tolerance during docking, as can be seen in Figure 2.5.

Figure 2.5: aPad-mounted docking mechanism comparison, initial design (left) and final design (right).

2.2.1 aPad guidance and control

A detailed examination and explanation of the dynamic and kinematic models of the platform, as well as developed low-level control structures, are given in ([59]). A brief overview is given here for completeness.

As a surface vehicle, of primary interest is the aPad's movement in a horizontal plane. The dynamic model can be described using a velocity vector alongside a vector of actuating forces and moments acting on the platform.

The velocity vector is given by $\boldsymbol{\nu} = \begin{bmatrix} u & v & r \end{bmatrix}^{\mathrm{T}}$ where u, v and r are the surge, sway, and yaw speeds, respectively.

The vector of actuating forces and moments acting on the platform is given by $\boldsymbol{\tau} = \begin{bmatrix} X & Y & N \end{bmatrix}^{\mathrm{T}}$ where X, Y are surge and sway forces and N is yaw moment. Both of these vectors are defined in the body–fixed (mobile) coordinate frame.

The platform is designed to be symmetrical with respect to the x and y axes in the body-fixed frame. Thus, the uncoupled dynamic model in the horizontal plane is given with (2.1) where $\mathbf{M}$ is a diagonal matrix with mass and added mass terms for each component expressed with $\mathbf{M} = diag\begin{pmatrix} \alpha_u, & \alpha_u, & \alpha_r \end{pmatrix}$, and $\mathbf{D}(\boldsymbol{\nu})$ is a diagonal matrix consisting of nonlinear hydrodynamic damping terms, component-wise $\mathbf{D}(\boldsymbol{\nu}) =$

$$diag\left(\begin{array}{ccc} \beta_u(u), & \beta_u(v), & \beta_r(r) \end{array} \right).$$

$$\mathbf{M}\dot{\boldsymbol{\nu}} = -\mathbf{D}(\boldsymbol{\nu}) + \boldsymbol{\tau} \tag{2.1}$$

The kinematic translational equations for the platform motion in the horizontal plane on the sea surface are given with (2.2).

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \underbrace{\begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix}}_{\mathbf{R}(\psi)} \begin{bmatrix} u \\ v \end{bmatrix} \tag{2.2}$$

Here x and y are the position and ψ is the orientation of the platform in the Earth–fixed coordinate frame, and $\mathbf{R}(\psi)$ is the rotation matrix.

An additional equation present in the kinematic model is $\dot{\psi} = r$. The platform is over-actuated, i.e., it can move in any direction in the horizontal plane by modifying its surge and sway speed, while attaining arbitrary orientation.

The low–level speed controller on the aPad is a PI controller was chosen, given with (2.3).

$$\boldsymbol{\tau} = \mathbf{K_{P\nu}}(\boldsymbol{\nu}^* - \tilde{\boldsymbol{\nu}}) + \mathbf{K_{I\nu}} \int (\boldsymbol{\nu}^* - \tilde{\boldsymbol{\nu}})\, \mathrm{d}t + \boldsymbol{\tau_F} \tag{2.3}$$

where $\boldsymbol{\nu}^* = \begin{bmatrix} u^* & v^* & r^* \end{bmatrix}^{\mathrm{T}}$ are the desired linear and angular speeds of the platform, and $\mathbf{K_{P\nu}} = diag\left(\begin{array}{ccc} K_{Pu}, & K_{Pv}, & K_{Pr} \end{array} \right)$ and $\mathbf{K_{I\nu}} = diag\left(\begin{array}{ccc} K_{Iu}, & K_{Iv}, & K_{Ir} \end{array} \right)$ are diagonal matrices with proportional and integral gains for individual degrees of freedom.

The values of the vehicle's speeds are often estimated (marked by a tilde sign in the equations), as they are either difficult to measure at all, or any available measurements are unreliable. The $\boldsymbol{\tau_F}$ term represents additional action introduced in the controller to improve the closed loop behaviour, as described in [60]. This action can be in the form $\boldsymbol{\tau_F} = \mathbf{D}(\boldsymbol{\nu})\,\boldsymbol{\nu}$ which results in the feedback linearisation procedure, in which measured or estimated speeds are used to compensate for the nonlinearity inherent in the process. It is more usual and convenient to use the feedforward term $\boldsymbol{\tau_F} = \mathbf{D}(\boldsymbol{\nu}^*)\,\boldsymbol{\nu}^*$. Controller parameters $\mathbf{K_{P\nu}}$ and $\mathbf{K_{I\nu}}$ can be calculated based on the desired closed loop characteristic equation as shown in [60].

Notable higher-level control primitives make use of the controllers described above. Primitives available on the aPads include a go-to-point manoeuvre, line following, dynamic positioning for station keeping, and a docking manoeuvre for collecting aMussels.

2.3 The aMussel robotic agent type

The aMussel (Figure 2.6) carries on it a wide selection of sensors, including temperature, pressure, turbidity, ambient light, and dissolved oxygen concentration. It has very limited movement capabilities, equipped only with a variable buoyancy system consisting of a piston and diaphragm which allows it to float to the water's surface, sink to the sea bed, or stay hovering at a set depth. For deployment and relocation to specific points in its environment, the aMussel requires the help of the aPad. Once deployed, the numerous aMussels use the miniaturised acoustic modems mounted on their top caps to function as an underwater acoustic sensor network (UASN) on a mission of long-term data collection and environmental observation. The aMussels employ acoustic communication with dedicated timeslots for each agent when underwater, and have WiFi, Bluetooth, and GSM capabilities when on the surface.

Figure 2.6: The aMussel underwater robot and sensor node. Note the inductive charging coils on the narrow neck below the top cap segment.

Initial deployment of the aMussel underwater sensor nodes within a chosen area of interest is also executed with the help of the aPad surface vehicles. Once released in the proper location, the aMussels will sink to the seabed, where they remain stationary while collecting data and occasionally communicate their findings to the surface, as well as amongst themselves. A variety of scenarios and behaviours is being developed and tested for the swarm agents, including trust and consensus-based decision making, deciding when to surface and request relocation, and how to determine and further explore points

of particular interest.

In order to make long-term autonomy possible, the aMussel was developed with low energy consumption in mind. Its main electronic board - called the MU (Measurement Unit) board - contains a Cypress PSoC4 microcontroller and is capable of deep hibernation, minimising energy consumption. The rest of its modules were developed and connected to the central board in a way that makes it possible to disable the power supply of each individual module and sensor as needed. The modules can be woken up by the main board, or by an acoustic signal received by the specialised miniature acoustic modem on their top cap. These configurable sleep cycles enable a considerable degree of adjustment - primarily reduction - of the robot's power consumption.

A lack of computational power is the downside of the designed aMussel main board and processing unit. As a potential workaround, each aMussel was also equipped with a Raspberry Pi unit with a custom-made adapter board and camera, making it possible to perform more complex and demanding calculations, image and on-line data processing, and store large amounts of data. The Pi board is only powered on in short intervals on rare occasions where its particular abilities are needed.

To further enhance autonomy, the aMussels are equipped with three inductive charging coils which enable battery recharging during mission execution. Each aPad has four matching mechanical docking stations housing inductive transmitter coils and guaranteeing good coil alignment, which allows it to charge up to four aMussels at a time. Energy exchange and deployment/relocation are among the key cooperative behaviours of the aMussel and aPad agents.

Wireless energy transfer is realised using a system based on inductive charging [61, 62], consisting of a power supply on the transmitter side, a set for wireless energy transfer, a battery charger, and a battery. The used wireless charging set is shown in Figure 2.7. It consists of a transmitter coil, connected to the the appropriate printed circuit board (PCB), which is connected to the power supply and receiver coil connected to the PCB, which is connected to the aMussel battery charger. Charging stops automatically when the wireless receiver stops transferring energy to the chargers or when batteries are full.

The aMussel itself houses two lithium polymer batteries and two independent battery chargers. Both batteries are charged simultaneously when in the charging dock: the primary battery is charged from two wireless receiver coils, while the secondary battery is charged from only one. The batteries are otherwise equivalent and equally capable and can be switched between at will using the power board of the aMussel, but in standard operation the primary battery powers the main board and all other operational boards and modules, while the secondary battery remains in standby and serves as backup. Monitoring of the charging process is accomplished by tracking the voltages and the

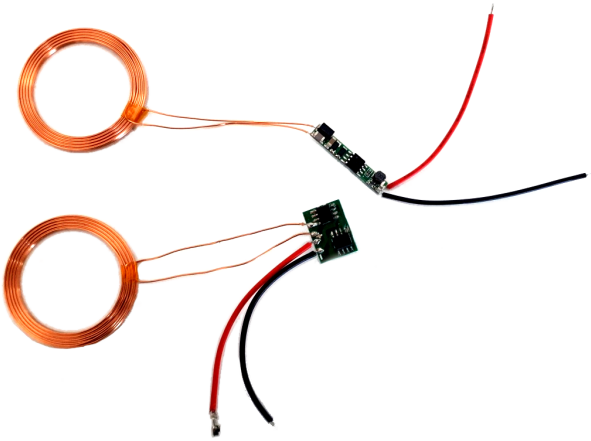


Figure 2.7: Wireless charging set present on each aMussel and aPad agent.

currents of the batteries, as well as the digital status pins of each battery.

The aMussel is passive agent during the docking process, conserving its battery and also easing unit production. Its top cap is covered with reflective tape which is saturated red when viewed using a normal RGB camera, and brightly reflective when viewed through an IR filter. Once it sends out its docking request and gets a confirmation of reception, an aMussel no longer has to keep any of its system powered, leading to energy conservation and fewer concerns about maintaining battery level above a certain threshold, as an aMussel can still be picked up by an aPad without issue even if its battery completely runs out. The aPad is the only active agent in the docking process, as both the IR emitter and receiver are entirely on its end.

2.4 Software architecture and simulation

2.4.1 Software architecture

The aPad's on-board vehicle software and the aMussel simulator software architecture is based on the Robot Operating System (ROS) middleware and its structure of nodes, services, and messages, realised using Python and C++. The overall structure can be seen in Fig. 2.8.

The simulation run on the surface station or on an aPad vehicle includes a ROS node containing a class representing a selected predefined number of aMussel agents with unique agent ID numbers. In its several modules it implements communication protocols, tracks agent positions and battery status, and provides simulated or dummy sensor data, depending on experimental scenario demands. The modules are all connected to the singular comms module and use the same simulation clock, but are otherwise unaffected by each other.

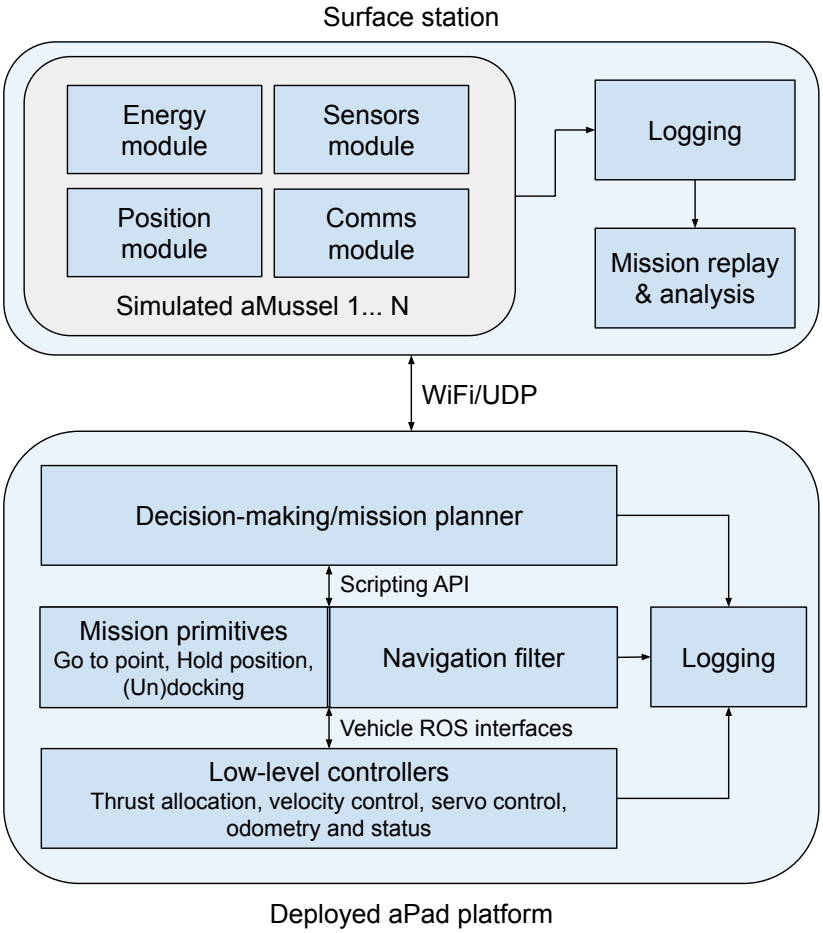


Figure 2.8: System software and communication structure - simulated aMussel agents and real aPad platforms.

The comms module subscribes to and parses messages received from aPads, and publishes aMussel sensor, position, and battery data at a fixed rate, using the same data structures and serialisation defined and used for real agents. The simulated aMussels can communicate with the aPads via WiFi (as if they are on the surface), while an interface for acoustic communication queuing exists, using either the “host" aPad’s nanomodem, or a separate physical nanomodem unit attached to the surface station computer via serial port. It is possible to intermix both real and simulated aMussels in one experiment, as long as care is taken to ensure no agent ID overlap.

The position module either generates a random uniform distribution of aMussels within a preset polygon representing the experimental area, or loads starting aMussel configurations from a file saved during a previous run, ensuring repeatability. A water current vector field map can be overlaid on the experimental area, causing agents to drift over time by translating their position each time step depending on the current vector. Depending on aPad status messages, the position simulation can switch between the free-floating mode and a charging mode in which aPad position is forwarded to the aMussel position

topic to represent the aMussel in question being docked.

The models used in the simulated battery module are described in detail in Subsection 2.4.2. Starting battery states can be randomly generated or loaded from a configuration file. The simulator charges and discharges all simulated aMussel batteries every fixed time step depending on charging status data received from the aPad via the comms module (i.e. responding to an alert once a certain aMussel has been “docked" by the aPad reaching its position and closing the designated docking mechanism, or “released" and thus no longer being charged).

Mission primitives implemented and used in the long-term monitoring mission include Go To Point for moving the aPad to a certain location, Hold Position for dynamic positioning, and Docking/Undocking for collecting and redeploying aMussels (described and discussed in detail later in this chapter). They are realised in the form of services that can be called manually or from within a mission script, accepting a set of parameters such as vehicle speed, victory radius, or docking mechanism selection. Controller parameters for all low-level controllers (e.g. yaw rate and surge speed for the docking procedure) can also be adjusted if needed.

As part of the overall system development, an Application Programming Interface (API) was developed to simplify high-level mission design and implementation. It is realised in the Python programming language and enables the creation (and automation thereof) of both extremely simple and linear task sequences as well as complex missions and behaviours, while removing the need for compilation before runtime. The API also contains definitions for all aPad- and aMussel-specific data structures. An example of a simple sequential aPad mission consisting of going to the last received position of the aMussel with the designated agent ID 24, then, once in position, initiating docking the mussel to the second docking mechanism, is given in Listing 2.1.

Listing 2.1: Example of scripting a simple aPad mission.

```
lat = api.get_location(24).lat
lon = api.get_location(24).lon
api.go_to([lat, lon])
while (api.goto_status(id_self) == 1):
print("GoTo in progress.")
time.sleep(1)
print("Goal reached. Docking start.")
api.docking(True, 2, 24)
```

The mission planner is realised using this scripting API. It contains a selection of strategies for collection and deployment which use a universal solution encoding format

in order to make it easy to swap between approaches and add new ones if desired. The mission planner also contains cost calculation modules with adjustable scale factors, making it possible to produce a variety of behaviours and observe their effects on aPad task sequencing. Solution representation takes the form of sequences of aMussel indexes, with the character "0" serving as a delimiter.

Data logging is realised in a redundant way: it takes place both locally on each aPad and on the surface station (if present). In an ideal case, the logs are equivalent and should loss of certain messages or data occur in any of them, the others can be used to fill in the gaps. The mission replay and analysis interface is realised in MATLAB based on the gathered *rosbag* format data logs (an example of the mission replay screen is given in Section 4.4). The Neptus C4I Framework [63] can also be used for aPad mission supervision, replay, and control, with aMussel data being relayed by connected aPads and displayed on the map overlay.

2.4.2 Battery modelling and simulation

During the fairly extensive subCULTron field tests, it was empirically determined that an aMussel was fully operational - meaning it could reliably turn on and use all of its modules at least once - in battery voltage ranges from a fully charged 4200 mV to a minimum of 3600 mV. For safety reasons, an aMussel should never be allowed to drop below this lower voltage limit, as this could mean it loses the ability to activate its buoyancy system motors and surface for recovery, leading to robot loss. The Raspberry Pi and the buoyancy system motors were additionally identified as the two largest power sinks present on the aMussel.

Fig. 2.9 (a) shows a realistic example of 20 hours of an aMussel operating overnight recorded in situ during one field experiment in Venice. The aMussel in this scenario sleeps and wakes up in regular intervals for several minutes in order to gather measurements, do some minor data processing, and await its designated acoustic timeslot, upon which it transmits an acoustic data packet - note the clearly visible voltage drops indicating brief hourly periods of increased activity interrupting low-power sleep mode. As contrast, the figure also shows an example of operation with unrealistically high power consumption, where an aMussel's Pi board was kept on, and its buoyancy motors ran every 10 minutes, leading to full battery discharge within 3 hours. For reference and comparison, recorded battery data of an aPad running its on-board computer and Kinect camera, activating all of its thrusters for 3 seconds every 30 seconds is shown in Fig. 2.9 (b).

In order to not only gather data for simulation design, but also test the long-term operating potential of the final developed aMussel system in a more structured manner, aMussel robots were placed on a laboratory table and left to discharge and then charged using aPad charging mechanisms at room temperature all while engaging in a variety of

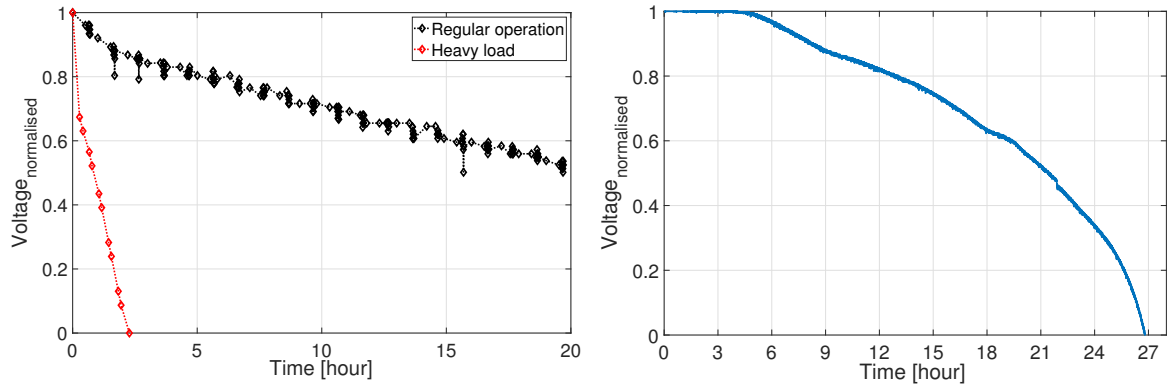


Figure 2.9: aMussel battery discharge data in two different modes of operation (a). aPad battery discharge scaled to operating range $[16, 10.5]V$ (b).

operating behaviours. Their battery voltage was measured, after which the data segment representing the previously determined permitted operational area of $[3600, 4200]$ was scaled to an interval of $[0, 1]$ to represent permitted operational limits.

The cases of particular interest for examining the discharging behaviour included a single aMussel battery discharging with the MU board turned on but idle (approximately 16 hours needed for full discharge) and the aMussel battery discharging with MU board in sleep, but waking up once every hour for about 10 seconds in order to record measurements from all sensors (approximately 9.8 days needed for full discharge).

Note that while discharging behaviours will be the same for both primary and secondary batteries under the same load, the charging will not due to the differing number of coils, so an additional charging use-case scenario was recorded. For the charging behaviours, of interest was the aMussel primary battery charging with the MU board turned on but idle (approximately 9.5 hours needed for full charge), the aMussel primary battery charging with sleep, waking up briefly in regular intervals to record sensor data (approximately 6.5 hours needed for full charge), and the aMussel secondary battery charging with the same sleep behaviour, with the MU powered by the primary battery (approximately 8 hours needed for full charge). The differences in durations of charging the primary and secondary battery are smaller than might be expected. For ease of comparison, recorded charging and discharging times are shown in Table 2.1.

The gathered data confirmed the long-term potential and viability of several aMussel use-cases that were established during the project. A sleeping aMussel waking up once an hour for a few seconds has enough time to retrieve measurements from all sensors, and the data it is monitoring consists of very slow-changing variables so sparse sampling is appropriate. Additionally, this awake period can be timed precisely to correspond to the aMussel's scheduled acoustic timeslot. In case of an emergency an aMussel that is asleep can be woken up by an acoustic ping addressed to it and receive whatever instruction is

Table 2.1: Times to full aMussel battery discharge and charge.

Discharging, MU board turned on but idle	962 min
Discharging, MU board in sleep, waking up once an hour	14057 min
Charging, primary battery, MU board turned on but idle	575 min
Charging, primary battery, waking up once an hour	398 min
Charging, secondary battery	485 min

necessary. Furthermore, while an aMussel is charging on the surface it is not in active use and can be asleep, speeding up the charging process. While taking into account the batteries' ageing and ability to hold charge [64], switching to and actively using the secondary battery should effectively double an aMussel's total deployment time.

These representative discharge and charge cycle datasets, as well as the simulation models of two specific behaviours that were chosen for the proof-of-concept experimental scenarios, are shown in Fig. 2.10 (discharging) and Fig. 2.11 (charging).

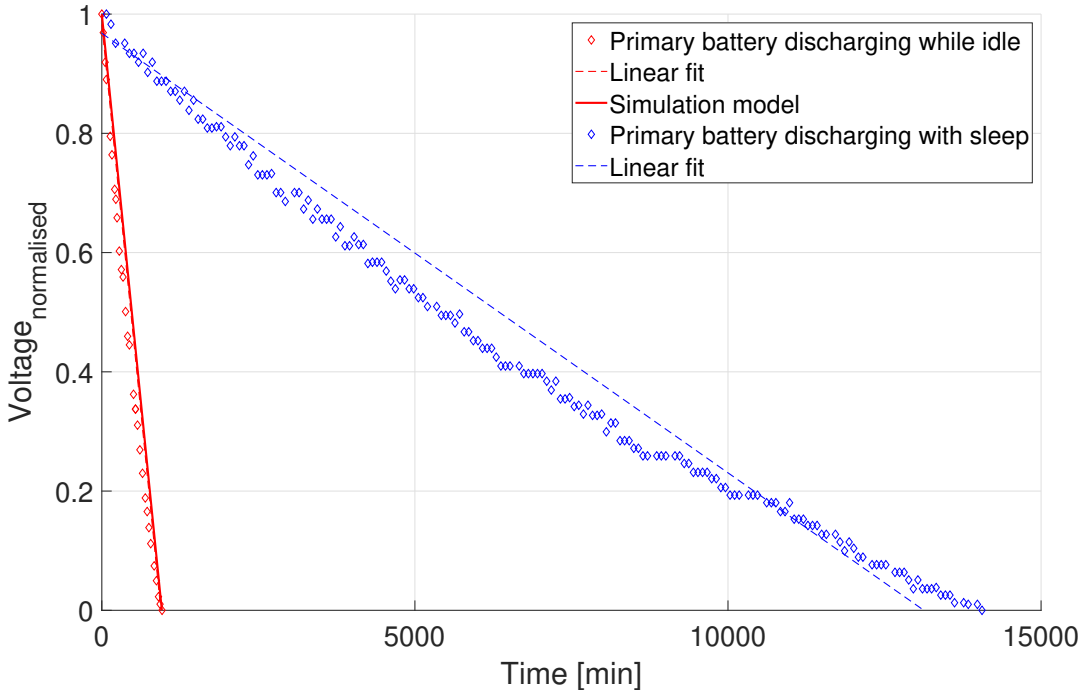


Figure 2.10: aMussel battery discharge data and implemented simulation model.

Least squares fitting was applied to the recorded data in order to get a simple discrete simulation model-suitable representation. A linear fit function was used for the discharging and an exponential represented using a piecewise-linear function for the charging. The equations were then implemented in a ROS node working at a rate of 1 Hz which

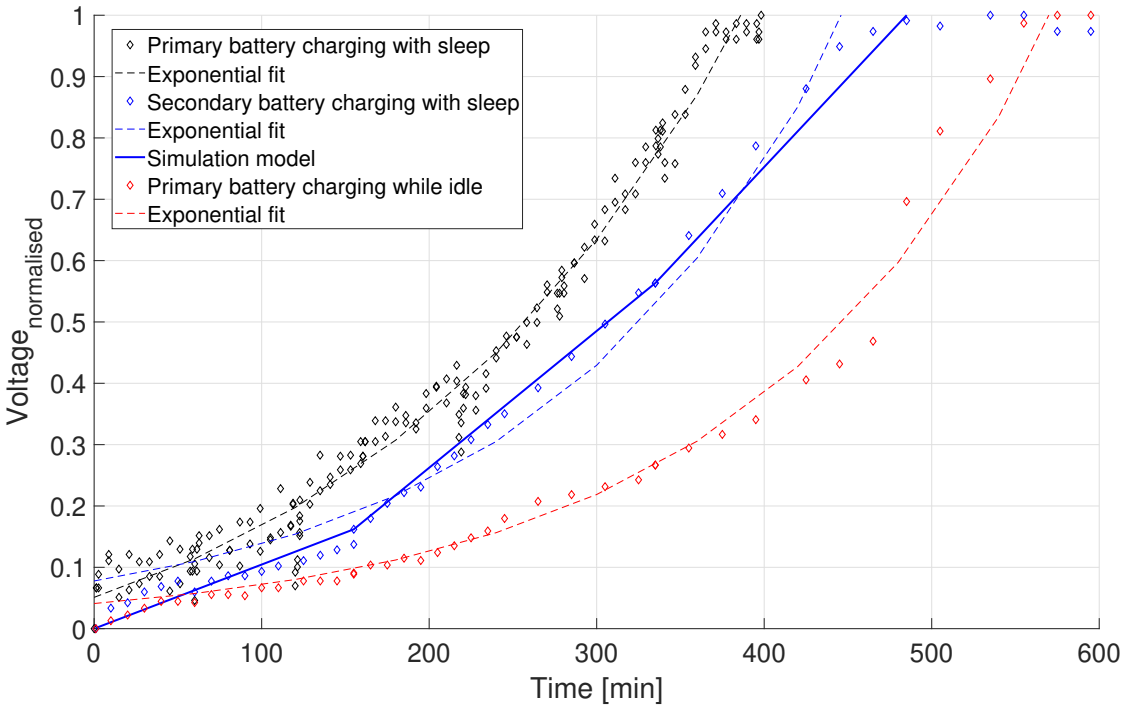


Figure 2.11: aMussel battery charge data and implemented simulation model.

updates and publishes current battery levels of an arbitrarily large group of aMussels, based on the last previously recorded battery levels and received charging status inputs for each simulated agent. Note that only voltage information was used in the simulation, as during simulation development remaining run-time indication was considered more important than any accurate state-of-charge modelling [65], and of primary concern was being able to make relative simulated aMussel battery state and charge/discharge time comparisons during high-level decision making. A time scaling factor was also introduced into the simulation, allowing for simulations where one unit of real time represents minutes or even hours in simulated time. The final selected simulation models used in the proof-of-concept scenario are, for discharging:

$$u_k = u_{k-1} - 0.00104271340 \cdot \Delta t \tag{2.4}$$

and for charging:

$$u_k = \begin{cases} u_{k-1} + 0.001045\theta \cdot \Delta t & u_{k-1} \leq 0.16197 \\ u_{k-1} + 0.00223\theta \cdot \Delta t & 0.16197 < u_{k-1} \leq 0.5634 \\ u_{k-1} + 0.002911\theta \cdot \Delta t & 0.5634 < u_{k-1} \end{cases} \tag{2.5}$$

Where u is voltage, k represents the simulation step and θ is the simulated time scaling factor. In this case with a 1 Hz sampling time, in real time the step is $\Delta t = 1s = \frac{1}{60}min$.

Final simulated voltage is clamped to $[0, 1]$.

2.5 Autonomous docking and energy exchange

The docking algorithm implemented in the marine robot swarm is an image-based visual servoing variant (IBVS) with end-point closed-loop control, since the visual sensor moves together with the effector towards the target, while itself integrated into the control loop of the system. The information about various offsets within the image acquired from the image processing segments of the algorithm is used directly in the control algorithm to calculate reference signals, as opposed to being used as a basis for estimating the real-world 3D space position or pose of the target. This is a classical, well-established and well-documented approach, as seen in [66], [67], [68]. More recently, applications of various implementations of visual servoing in the field of marine robotics can be seen in [69], where a visual servo control approach was successfully transferred from an industrial manufacturing context to underwater robotics tasks autonomously performed by a subsea hydraulic manipulator mounted on a work-class Remotely Operated underwater Vehicle (ROV). In [70], visual servoing is used as a method of achieving station-keeping in an underwater environment, with unmarked natural features acting as targets for the image tracking algorithm and enabling an unmanned underwater vehicle (UUV) to hover above planar targets on the sea bed. In [71], a more complex hierarchical control structure featuring visual servoing is introduced in order to achieve dynamic positioning of a fully actuated underwater vehicle.

2.5.1 Docking algorithm

The control structure of the systems involved in the docking algorithm consists of the following components: Docking phases that contain a state machine that implements the highest level of control present in this system; Guidance and control algorithms for generating references for the aPad's low-level speed controllers; and Image processing, which provides information and feedback from the aPad's visual sensors necessary for the functioning of the docking algorithm by employing traditional computer vision methods and neural network object detection, then fusing them using an Extended Kalman filter. This structure is shown in Figure 2.12.

Details of the dynamic and kinematic models of the platform, as well as developed high- and low-level control structures, are given in [59].

A state machine representation of the high-level control of the aPad docking algorithm is given in Figure 2.13.

The aPad docking control algorithm loop consists of three active phases:

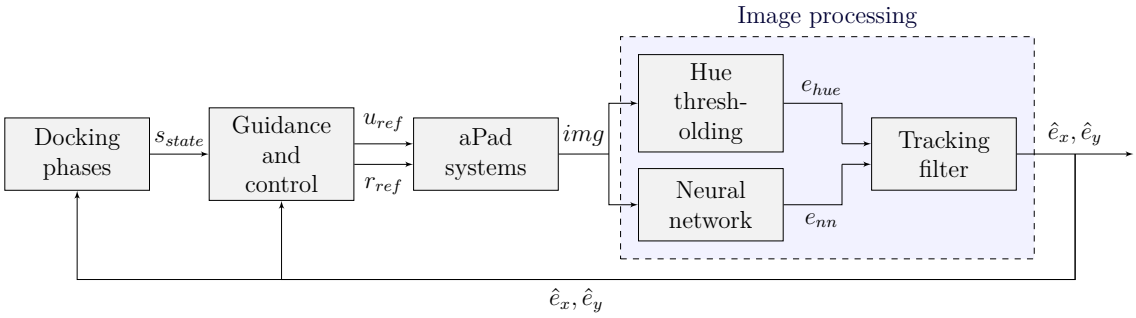


Figure 2.12: Control structure for autonomous docking implemented on each aPad.

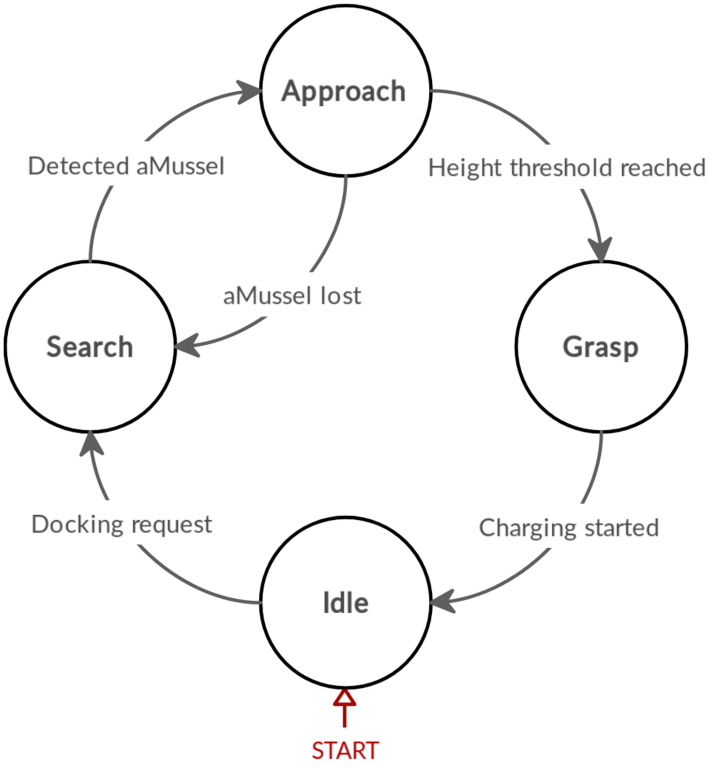


Figure 2.13: State machine representation of the high-level aPad controller running the docking algorithm.

- **Search** - The aPad rotates at a set rate until it registers a camera frame in which the top cap of an aMussel is visible.
- **Approach** - The aPad moves towards the detected aMussel, turning in order to keep it centred in view and thus properly aligned with the mechanism for docking. Should the aMussel be lost from view during this phase, the aPad will recommence Search.
- **Grasp** - The aPad closes the servo-actuated gripper on its docking mechanism once the reported position of the aMussel is close enough, i.e. below the set height threshold. Charging is started.

An **Idle** state is also present to account for the times when the aPad is holding its

position and allowing the docked aMussel to charge and/or transmit data. Before each docking attempt starts (launching into the **Search** state), the pan mechanism rotates the Kinect sensor to face the chosen dock.

The gripper on the docking mechanism is closed once the y coordinate of the located aMussel top cap falls below a vertical threshold, indicating that the mussel is close to the camera as well as the docking mechanism it is pointing towards. This value has been calibrated with regards to the camera angle and the buoyancy and height of the fully surfaced aMussel in order to ensure a timely closing, and is currently set to 25% of the full image height.

During the Approach phase of the docking algorithm, the aPad moves towards the detected aMussel, regulating its surge speed (and thus the speed of its forward approach to the aMussel) using the curve given in Figure 2.14 - the closer the position of the aMussel is to the centre of the image, the faster the aPad will approach it. Yaw rate is regulated using the curve shown in Figure 2.15. Thus, if the aMussel is far off to the side of the aPad's current view, the emphasis will be on yaw movement as opposed to surge movement, and the aMussel will be kept aligned with the docking mechanism.

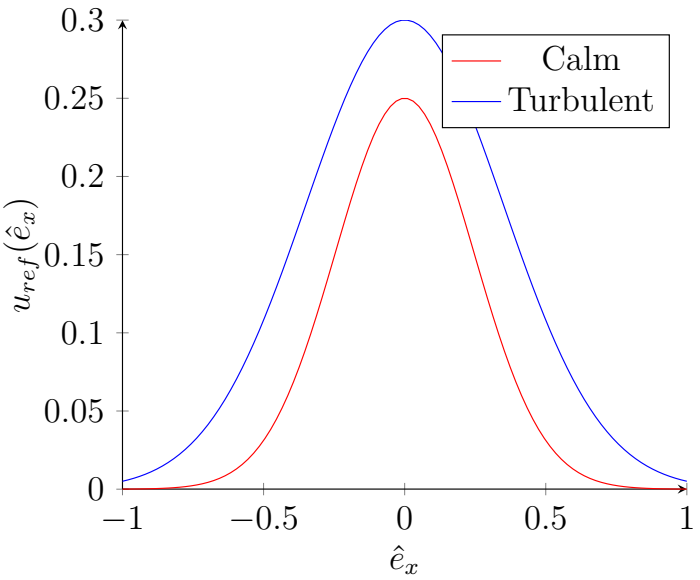


Figure 2.14: Surge speed curve used during autonomous docking. $\hat{e}_x$ is the estimate of the x or horizontal image coordinate of the aMussel.

The equation used for generating references for surge speed control is:

$$u_{ref}(\hat{e}_x) = k_1 e^{-\frac{\hat{e}_x^2}{2\sigma^2}} \tag{2.6}$$

and the equation used for yaw rate control is:

$$r_{ref}(\hat{e}_x) = k_2 \tanh(a\hat{e}_x) \tag{2.7}$$

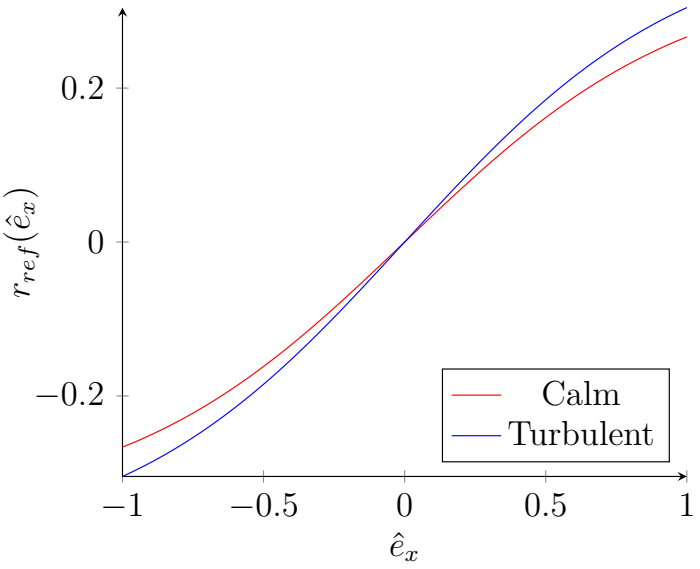


Figure 2.15: Yaw speed curve used during autonomous docking. $\hat{e}_x$ is the estimate of the x or horizontal image coordinate of the aMussel.

Table 2.2: Regulator parameters

Parameter	Value - calm	Value - turbulent
k_1	0.25	0.3
k_2	0.35	0.4
σ	0.245	0.35
a	1	1
$r_{ref,search}$	0.3	0.4

where the input $\hat{e}_x$ represents the estimate of the horizontal image coordinate of the aMussel scaled to an interval of [-1,1], received from the filter node. A fixed yaw rate $r_{ref,search}$ is set for the search phase of the docking process. Values chosen for regulator parameters for working in calm water conditions and for working in a more turbulent environment are given in Table 2.2.

Depending on which of the four aPad docks the aMussel is being docked to (or undocked from), the surge reference needs further transformation. A rotation matrix is applied to the generated surge reference from (2.6):

$$\boldsymbol{u}'_{ref} = \begin{bmatrix} u'_{ref,x} \\ u'_{ref,y} \end{bmatrix} = \begin{bmatrix} cos(\alpha) & -sin(\alpha) \\ sin(\alpha) & cos(\alpha) \end{bmatrix} \boldsymbol{u}_{ref,0} \tag{2.8}$$

where $u_{ref,0}$ represents the reference for the default front-facing dock which is used to

determine the vehicle's heading:

$$\boldsymbol{u}_{ref,0} = \begin{bmatrix} u_{ref} \\ 0 \end{bmatrix} \tag{2.9}$$

Angle α is simply determined using $\alpha = i_{dock}\frac{\pi}{2}$, with i_{dock} as the index of the chosen dock, as shown in Figure 2.16.

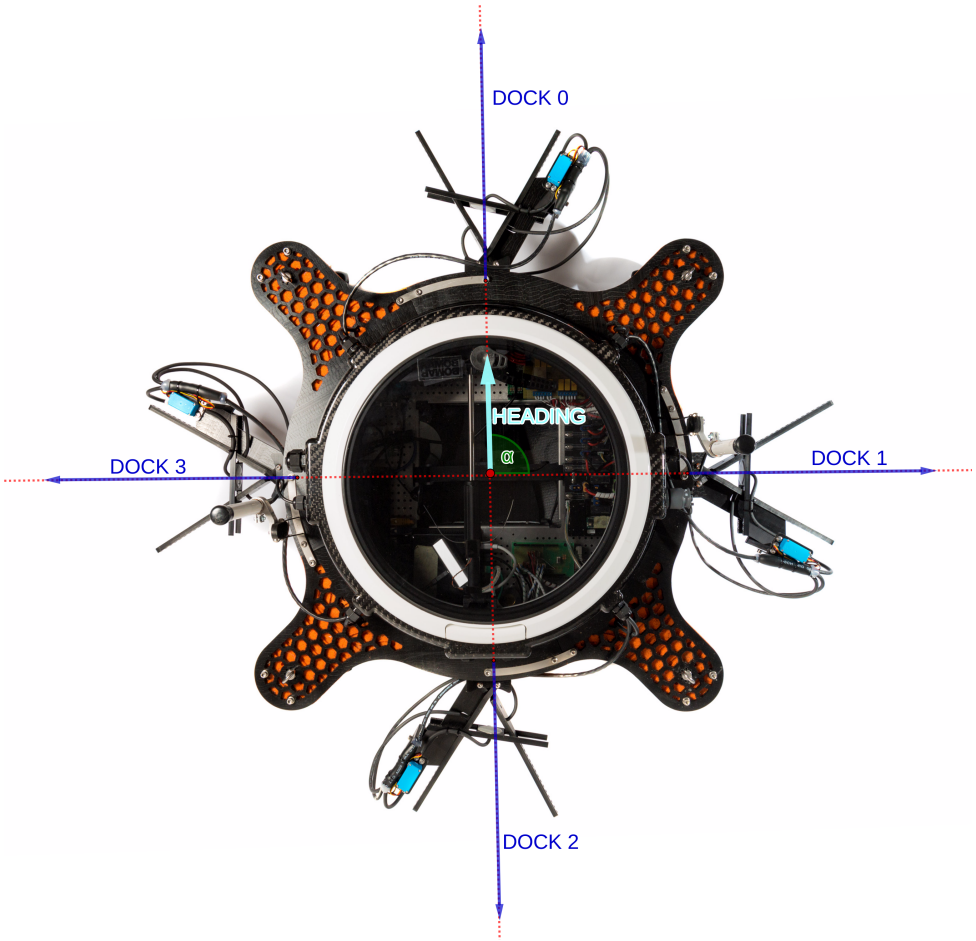


Figure 2.16: aPad vehicle viewed from above with docks marked in indexing order. Example angle α highlighted for dock with index 1.

2.5.2 Image processing

Achieving a good degree of accuracy in image processing - especially when it is used as one of the main methods of robot perception - is key in real-world applications. The image processing approaches used for locating the aMussel and correctly navigating towards it evolved through several stages that roughly correspond to the evolution of the system hardware. These include:

- IR-only intensity thresholding
- Hue-based thresholding

- Neural network detection

Despite the evolution of the system, the option to use each of the approaches individually remains. The ability to switch between ("legacy") IR mode and the more recent full RGB image modes is of practical significance, as this duality means swarm work in dark or night-time conditions can still proceed. The capability of the Kinect sensor to switch between infrared and regular camera is one of the main reasons it was chosen. This section describes the implementation of the three image processing variants.

Note that although the Kinect sensor is mounted sideways, all images are processed without rotation. Rotation is done later in coordinate calculations before feeding information to the parts of the system which formulate aPad movement references, which saves time in potentially costly image processing operations. For ease of viewing and reading, the images here presented as examples have been rotated.

2.5.3 IR-only intensity thresholding

To begin the infrared based detection, a greyscale image is retrieved from either the initially used analog camera or the Kinect's infrared sensor, and a mild low-pass Gaussian blur filter is applied to it in order to reduce noise. The position of the pixel with the highest value found in the image is determined (as higher value means lighter pixels, which is convenient for locating light sources in images).

After this, the image is thresholded so that only the brightest cluster of pixels remains in it - the adaptive threshold applied to the image is set using the maximum pixel value determined in the previous step. Next, a bounding box is placed around the largest cluster of pixels remaining in the image after thresholding. Visual servoing is done by passing the coordinates of this bounding box to the controller, with the assumption that the origin of the coordinate system is in the centre of the image, both vertically and horizontally. Coordinates are scaled to a range of [-1,1] for control purposes, with -1 corresponding to the left edge and bottom of the image, and 1 being the right edge and top.

An example comparison of raw camera input and processed image taken during one of the indoor pool tests (with the coordinate system overlaid for reference) is given in Figure 2.17.

This approach to visual servoing was used as the primary method during indoor testing of the system and the initial docking experiments, as outlined in [45]. Its range proved limited, however, and certain components such as the IR LEDs in the aMussel which served as beacons for detection were removed from the system after the initial prototyping stages.

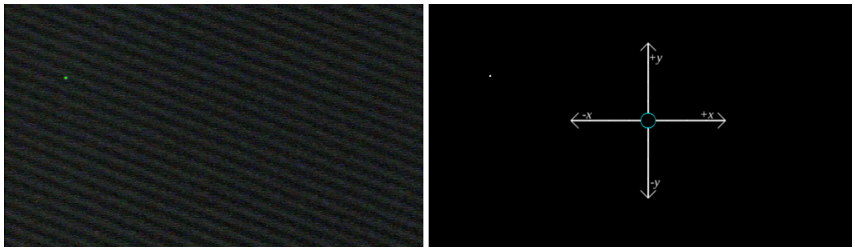


Figure 2.17: Original analog camera image (left) and processed image with coordinate system (right), showing a view of the experiment testbed with present IR LED in aMussel cap. The green rectangle overlaid on the left camera image represents a bounding box around the final derived position of the IR LED used to control the aPad's approach.

2.5.4 Hue-based thresholding

The original implementation of the hue-based thresholding method for detecting and locating aMussels in an image was described in [45].

The algorithm first crops the RGB image received from the Kinect sensor to a region of interest representing a fixed distance in front of the camera in which an aMussel might actually be located. This serves to both minimise the chances of false positive detections and not waste processing time on a region of image that is likely to contain only sky. Currently the crop disregards the upper 25% of the 640 pixels high image. The cropped image is then converted into the HSV (Hue Saturation Value) colour space, and Contrast-limited Adaptive Histogram Equalization (CLAHE) is applied to it in order to improve robustness of detection in varying weather conditions (for examples of using CLAHE to improve visibility in noisy conditions and poor lighting, see [72] and [73]). Finally, a threshold based on hue values for the colour red is applied to the image.

Similar to the IR thresholding method, a bounding box is placed around the largest cluster of pixels remaining in the image after thresholding. Visual servoing is again done by passing the coordinates of this bounding box to the controller, with the same assumptions about the image coordinate system. Two examples of aMussels detected in the processed camera image are given in Figure 2.18.

Several heuristics to improve detection have been implemented. There is a minimum blob size requirement for detection in order to remove noisy false positive detections of reflections, floating debris or similar. Primarily, the size of the pixel blob and its position in the image are considered: for example, if a very large blob is positioned higher in the image, it is highly unlikely to be an aMussel, and will be disregarded.

As noted in [45] and included here briefly for completion, an analysis of image data collected on-site and processed with the developed hue-based aMussel detection algorithm was compared to manually annotated data in order to achieve a benchmark. The results are shown in Table 2.3 and Table 2.4.

Figure 2.18: Two examples of aMussel cap detection - original images shown left and processed images shown right. The green rectangle on the camera image represents a bounding box around the derived position of the cap used to control the aPad's approach.

Table 2.3: Comparison of frame counts with and without aMussels present.

Manually annotated data	No mussel	1723
	Mussel present	1098
Automatically processed data	No mussel	1847
	Mussel present	974

Table 2.4: Frame count breakdown for aMussel detection algorithm.

Correctly detected empty frames	1700
False positive frames	23
Missed frames	147
Correctly detected non-empty frames	951
Total frames processed	2821
Success rate	86.61202%

2.5.5 Neural networks for object detection

In an attempt to make the aMussel detector more universal, less reliant on lighting conditions and needing fewer hue threshold adjustments before deployment, as well as to increase maximum detection distance and generally increase robustness, the decision was made to use artificial neural networks trained for object detection and recognition.

The neural network training was done on a PC with two Nvidia GTX 1080 Ti graphical processing units, running the Nvidia CUDA toolkit to achieve parallel computing. The training process was further sped up using the OpenCL framework which enabled the learning to also be executed on the CPU - in this case an Intel Core i9 9900k with 8 cores and 16 threads running at frequencies up to 5 GHz.

A dataset consisting of 3105 images containing an aMussel was constructed from a mixture of data collected in several locations and under varying conditions at Jarun lake in Zagreb, Croatia, the seaside at Biograd na Moru, Croatia, and the Arsenale and lagoon of Venice, Italy. Object detection and recognition networks were trained with one detection class in mind - the *amussel* class, using the Tensorflow library [74]. Dataset annotation in the form of labelled bounding boxes was done manually using the open source LabelImg tool [75].

Several popular object recognition models were trained and tested, in the interest of finding the best one for the specific application. The selected models are the Single Shot Detector with MobileNet version 2 (SSD MobileNetV2) [76] trained in two instances for two different input image sizes (a small and fast 300x300 resized version, and a 640x480 version without resizing), and the Fast Region-based Convolutional Network method (FR-CNN) [77]. Additionally, a You Only Look Once version 3 detector (YOLOv3) was trained on the data using its own specific training pipeline [78]. An example comparison of the neural network output compared to the labelled “ground truth" it is learning from can be seen in Figure 2.19.

Figure 2.19: Detection (left) vs ground truth (right) mid-training example for the SSD Mobilenet V2 model.

On the dataset of 3105 images, a 90/10 train/test split was used, and validation was later additionally run on an entirely fresh unseen dataset. Testing the final frozen graphs on both train and test data yielded only a marginal decrease of accuracy on the previously unseen test-exclusive split of the dataset, implying that no significant overtraining occurred and the networks were appropriate for further testing and use. Some examples of detection in challenging circumstances are shown in Figure 2.20.

Figure 2.20: Difficult detection examples (output aMussel bounding boxes shown in blue) which neural networks were extremely helpful in resolving - including cases with larger distances, partial visibility, occlusion, wave splashes/partial submersion, bad lighting and weather conditions, and sunlight glare.

Striking a balance between detection reliability and framerate on the limited computer running within the aPad was a significant part of the task. Framerates were tested on the Intel NUC i5 on-board computer on the marine platform, running in parallel with the control structures of the aPad, as these are the operating conditions relevant to the

Table 2.5: Comparison of mean average precision in single-class object detection and single image frame processing time on on-board computer (bold is best, italic is final choice).

Model	mAP	Seconds per frame (average)
SSD Mobilenet v2 300x300	0.6452	**0.1**
SSD Mobilenet v2 640x480	*0.7064*	*0.2*
Faster R-CNN	0.7375	1.6
YOLOv3	**0.9352**	2.7

application. Performance results are shown in Table 2.5, where mean average precision (commonly known as mAP) was used as a measure of model reliability and object detection quality [79].

YOLOv3 offered excellent accuracy, but was significantly computationally demanding (especially since there is no dedicated GPU present on the aPad) and thus too slow when running in real time. After testing and benchmarking, as the best compromise between detection quality and reliability and framerate, the SSDN MobileNet V2 with no additional image resizing was chosen, but a ROI crop was applied to the input image, same as in the hue thresholding method.

Running in parallel, the original hue thresholding algorithm is capable of outputting processed images at a rate of 10 FPS, meaning the neural network will be made to process only the latest image it has in its input queue, potentially skipping some older enqueued image frames in order to not lag behind significantly enough to affect the docking process. The outputs of the hue thresholding and the neural network detection are then fused together using a filter, which determines the final aMussel position estimate.

The neural networks can easily find multiple aMussels in a single frame. Traditional processing could be made to manage this too, but it would impact reliability of results. Of course, only one aMussel at a time can be in the process of being docked, so this does not impact the docking procedure in a major way.

2.5.6 Tracking filter

In order to combine the output of the classical hue thresholding approach (which had proven quite reliable in close quarters) and the neural network approach to image processing, and to use the benefits of both while compensating for the drawbacks of each, an extended Kalman filter (EKF) was introduced into the system. The filter operates in the image coordinate space, and fuses together aMussel position measurements. It also acts as a tracking filter, using previously acquired information about the speed of the aMussel's movement in the image to ensure more accurate state predictions.

The filter operates at a fixed rate of 10Hz, so a time step of $\Delta t = 0.1$s is used in all internal calculations.

The state vector of the filter is given with

$$\mathbf{x} = \begin{bmatrix} \hat{e}_x & \hat{e}_y & v_x & v_y \end{bmatrix}^T \tag{2.10}$$

where v_x and v_y are speeds, and $\hat{e}_x$ and $\hat{e}_y$ are position estimates based on measurements fused from the two sources. The state model used can be expressed with $\dot{\hat{\mathbf{e}}} = \begin{bmatrix} v_x & v_y & 0 & 0 \end{bmatrix}^T$.

The measurement vector of the filter is given with

$$\mathbf{y} = \begin{bmatrix} e_x & e_y \end{bmatrix}^T + \begin{bmatrix} r_x & r_y \end{bmatrix}^T \tag{2.11}$$

where e_x and e_y represent (x, y) image coordinate pairs calculated using measurement fusion and scaled from pixel values to a [-1,1] range.

Results received from the two measurement sources are represented with $\mathbf{e_{nn}} = \begin{bmatrix} e_{nn,x} & e_{nn,y} \end{bmatrix}^T$ containing the results of neural net processing and $\mathbf{e_{hue}} = \begin{bmatrix} e_{hue,x} & e_{hue,y} \end{bmatrix}^T$ containing the results of hue-based image thresholding.

Meanwhile, r_x and r_y are the respective measurement variances used in order to take into account measurement quality and reliability along each axis. These are defined as:

$$r_x \sim \mathcal{N}(0, \zeta_x) \tag{2.12}$$

$$r_y \sim \mathcal{N}(0, \zeta_y) \tag{2.13}$$

The values of ζ_x and ζ_y are determined using values for each measurement source, $\boldsymbol{\zeta_{nn}} = \begin{bmatrix} \zeta_{nn,x} & \zeta_{nn,y} \end{bmatrix}^T$

for neural net processing and $\boldsymbol{\zeta_{hue}} = \begin{bmatrix} \zeta_{hue,x} & \zeta_{hue,y} \end{bmatrix}^T$ for hue-based image thresholding.

For neural network data, the values of $\boldsymbol{\zeta_{nn}}$ are set to a constant:

$$\zeta_{nn,x} = \zeta_{nn,y} = 0.075 \tag{2.14}$$

Whereas for the hue thresholded data, the $\boldsymbol{\zeta_{hue}}$ values were defined as depending on the vertical position of the detected pixel blob in the image. The further away on the y-axis the detected blob is, the higher the variance of the hue-based detector, i.e. hue-based detection is less trusted. It is set with:

$$\zeta_{hue,x} = \zeta_{hue,y} = \frac{1 - e_{hue,y} + 0.2}{2} \tag{2.15}$$

Considering the image y-coordinate ranges from -1 at the top of the image, and 1 at the bottom, the variance values used in the filter achieve a range of $[0.1, 1.1]$, which is always higher than any concurrently available neural network detector value.

Thus, using methods explained in [80] and [81], the fusion of measurements from two sources is done with:

$$e_x = \frac{\zeta_{hue,x} e_{nn,x} + \zeta_{nn,x} e_{hue,x}}{\zeta_{hue,x} + \zeta_{nn,x}} \tag{2.16}$$

$$e_y = \frac{\zeta_{hue,y} e_{nn,y} + \zeta_{nn,y} e_{hue,y}}{\zeta_{hue,y} + \zeta_{nn,y}} \tag{2.17}$$

$$\zeta_x = \frac{\zeta_{hue,x} \zeta_{nn,x}}{\zeta_{hue,x} + \zeta_{nn,x}} \tag{2.18}$$

$$\zeta_y = \frac{\zeta_{hue,y} \zeta_{nn,y}}{\zeta_{hue,y} + \zeta_{nn,y}} \tag{2.19}$$

Implementation-wise, three distinct cases exist. If hue-based detection has produced a result, but no detection data has been received from the neural net, $\boldsymbol{\zeta_{nn}}$ values will both become zero, while $\boldsymbol{\zeta_{hue}}$ will be calculated using (2.15). The fused measurements e_x and e_y will simply become $e_{hue,x}$ and $e_{hue,y}$, respectively, while ζ_x and ζ_y will become $\zeta_{hue,x}$ and $\zeta_{hue,y}$.

Likewise, if no detection data has been received from the hue processing node while a valid result has been given by the neural net, $\boldsymbol{\zeta_{hue}}$ will be a zero-vector and $\boldsymbol{\zeta_{nn}}$ will be calculated using (2.14), with the fused measurements e_x and e_y respectively becoming $e_{nn,x}$ and $e_{nn,y}$, and ζ_x and ζ_y becoming $\zeta_{nn,x}$ and $\zeta_{nn,y}$.

Finally, if both nodes have sent valid object detection data, their respective variances and fused measurements are calculated as has been described above in (2.14), (2.15), (2.16), (2.17), (2.18), and (2.19).

Figure 2.21 shows a comparison of the hue thresholding result, the neural network result, and the final filter output during one segment of a successful docking experiment. Docking occurs around 130s, undocking around 145s. Note the way the hue thresholding comes into play as the aMussel approaches the camera, and the filter seamlessly segueing between the two sources.

In addition to distance-based variance calculation, the filter node contains several heuristics aimed at improving data fusion and detector behaviour:

- *Kinect pan movement flagging.* The filter will not take into account measurements acquired while the Kinect sensor was actively rotating, and will only react once a valid docking mechanism-aligned position has been reached.

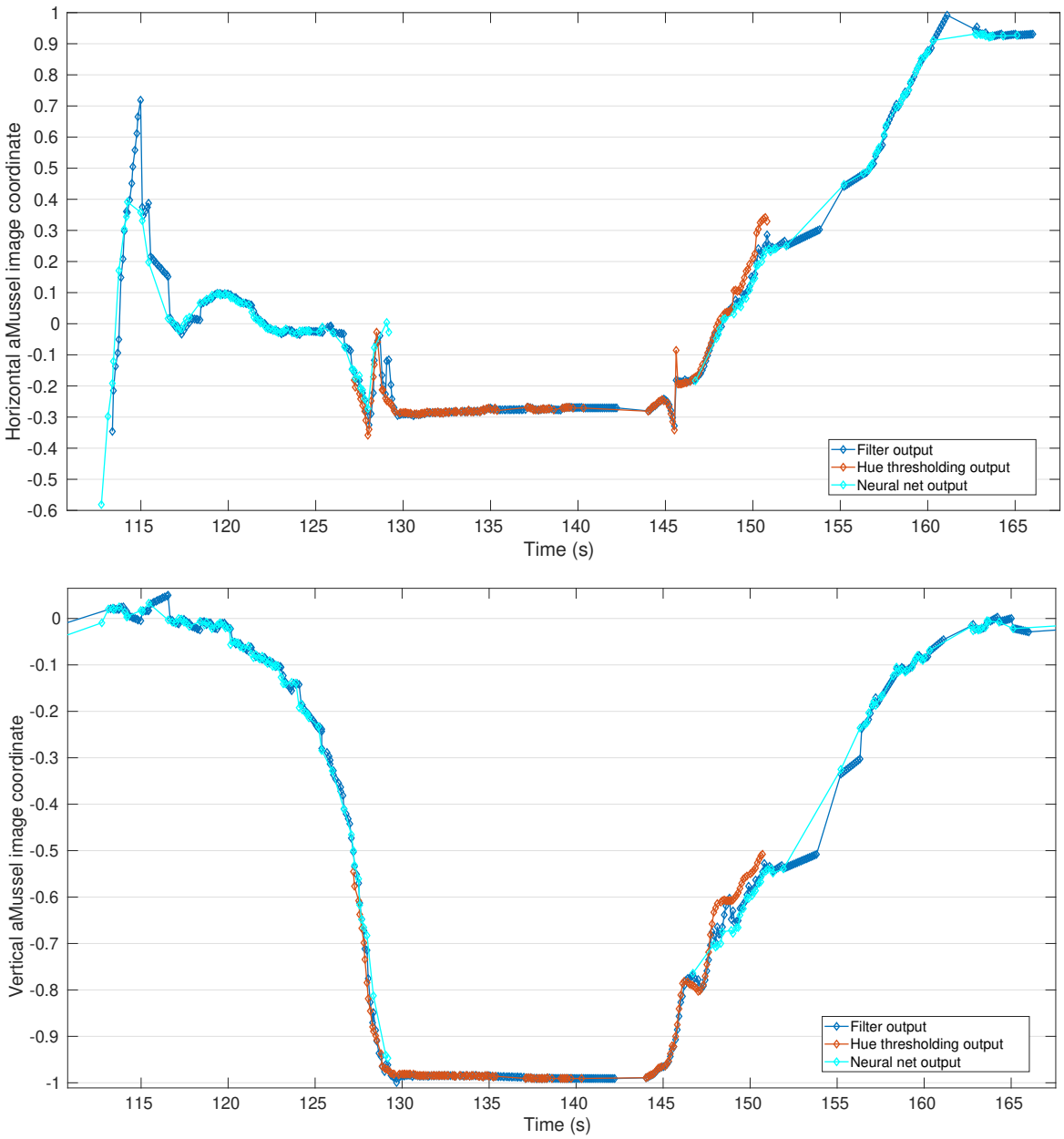


Figure 2.21: Example of filter inputs and output during a single docking, with markers denoting individual measurements where received, horizontal (above) and vertical (below).

- *Region of interest crop.* As the filter converges, any filter outputs outside of the designated realistic “region of interest" (-1 to 1) are discarded.
- *Continuous detection requirement.* The filter will only output fused measurements after a preset number of consecutive frames have been determined to contain an aMussel, by any of the two detectors. This acts as a sort of outlier rejection for brief usually single-frame false-positive detections.
- *Measurement discontinuation detection.* If more than 3 seconds have passed without fresh measurements from either detector, the filter will stop feeding measurements into the docking controller, and the filter will be flushed and reinitialised. This en-

sures no action is taken based on outdated information. If the aPad was approaching an aMussel and lost it from view this long, as per the previously described state machine the docking algorithm will return to the Search state.

2.6 Experimental validation of autonomous docking

The subCULTron swarm has in the four and a half years of the project's runtime been through many tests and demonstrations, including numerous field deployments and experimental trials. The docking segment in its various incarnations has been included and tested throughout. Presented in this section is a detailed writeup of three particular experiments: initial indoor pool experiments performed in Brodarski Institute in Zagreb; a structured experiment to determine the capability of and validate the final docking procedure undertaken at lake Jarun in Zagreb, Croatia; and a “stress test" in a challenging environment on-site near the island of Sant'Angelo della Polvere in the lagoon of Venice, Italy.

2.6.1 Indoor pool experiments

In [45] the indoor experiments which served as proof of concept for the wireless charging are described, as well as initial outdoor experiments aimed at testing the capabilities of the aMussel and aPad positioning systems in the interest of combining them and including them in the docking procedure.

Initial indoor experiments were done at the large circular pool of the Brodarski Institute in Zagreb. The testbed is shown in Figure 2.22. The aPad used in these early experiments had only one docking mechanism mounted and the infra-red mode of detection was used.

The experiment procedure was as follows:

1. aMussel floating on the surface requests charging.
2. Nearby aPad autonomously searches for aMussel using camera.
3. Once located based on image data, aPad approaches the aMussel.
4. Once close to the aMussel, the aPad grabs it and charging starts.
5. aMussel reads charging status from its power board at 1Hz and broadcasts received data via WiFi.
6. Data is displayed and logged on the surface station.
7. Once the aMussel has been released and the charging status has returned to 0 (no charging occurring), it shuts off data broadcasting after a timeout and sinks to the bottom.

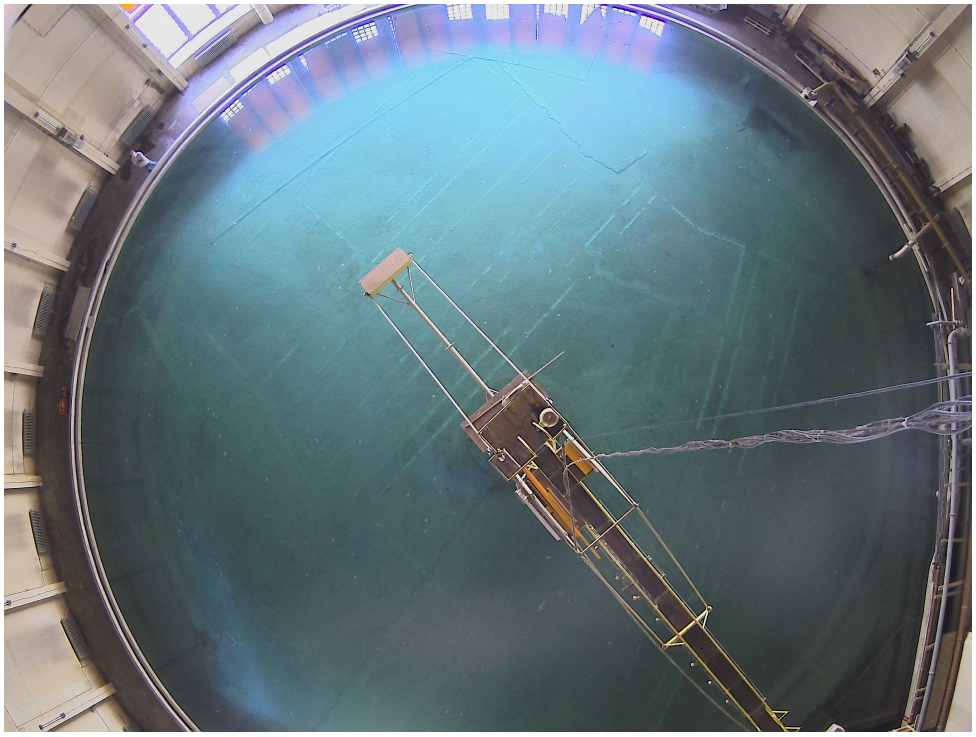


Figure 2.22: Top-down view of pool used for initial indoor experiments.

A short video overview of indoor autonomous docking experiments can be found at the following link: `https://www.youtube.com/watch?v=fgZDF3tGIVY`.

Due to the lack of an indoor localisation system, before the start of each experiment run the aPad was manually positioned to be within 2-3 metres of the aMussel. Starting distances above 3.5 metres made the docking procedure unreliable since they proved challenging for the visual servoing system due to the small size of the target.

The starting rotation of the aPad relative to the aMussel was changed in each experimental run, including edge cases such as the aMussel being very close to the aPad but on the opposite side from the camera. Starting rotation variations did not prove to have a noticeable effect on the reliability of the autonomous docking, with one noticeable difference between the cases being the length of the first search phase. Charging status and current measured from both the batteries present in the aMussel for the duration of one docking experiment are shown in Figure 2.23.

Negative measured current signifies that the battery is receiving charge, making the start and end of charging easily visible in the resulting plots. Charging status is an indicator which is reported as 0 when there is no charging, and 1 when charging is happening. As expected, the measured current jumps into positive values as soon as charging is stopped. As noted earlier, primary battery A receives more charge than backup battery B (its current measurements reach larger negative values: -750mA average as opposed to -300mA average), since it is connected to two of the three inductive coils.

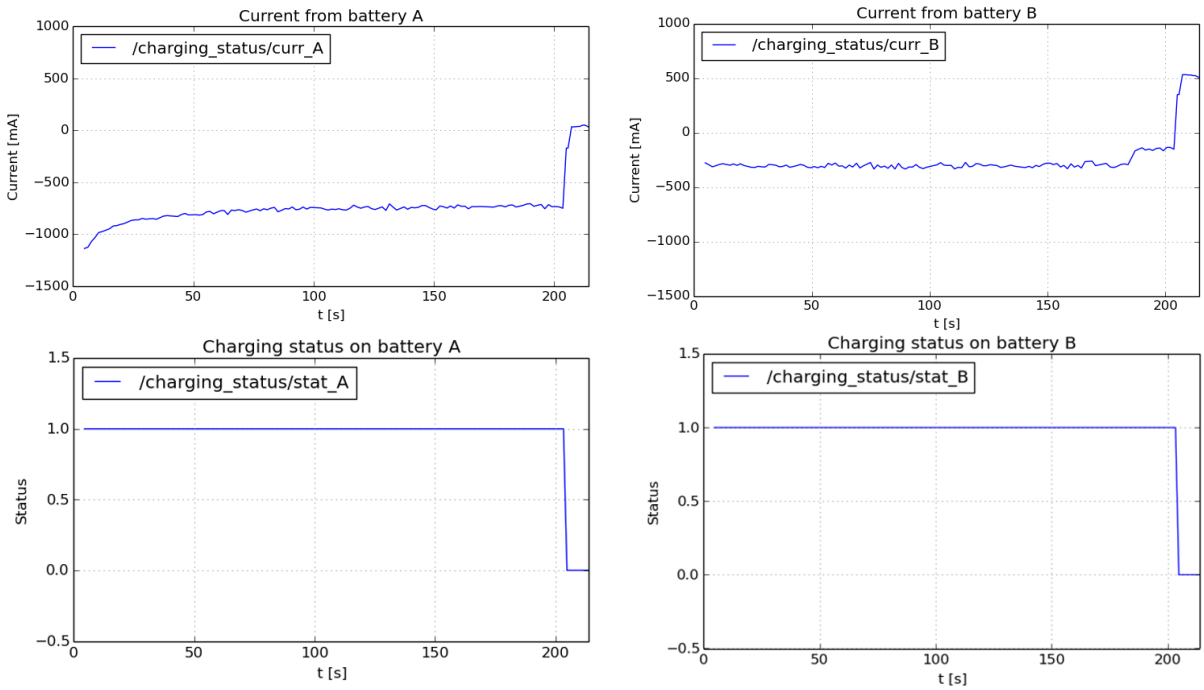


Figure 2.23: Battery current and charging status of batteries A (a) and B (b).

2.6.2 Initial outdoor experiments

Initial outdoor experiments were conducted in Venice, Italy and in Biograd na Moru, Croatia. The goal was to test both the visual servoing and the entire previously described docking procedure, including WiFi communication between the two types of agents, in varying real-world conditions, including strong wind and current (Figure 2.24).

Figure 2.24: aPad with pan mechanism and Kinect sensor docking an aMussel (red cap, left, partially submerged) during outdoor tests.

A short video overview of outdoor autonomous docking experiments can be found at the following link: `https://www.youtube.com/watch?v=0Ao92MF0HDo`.

Figure 2.25 shows the course of one run of the docking procedure experiment. The position and heading of the aPad are visible. Green segments represent the aPad moving from point to point. Red segments denote the aPad going through the search and approach docking phases, or, if there a mussel already docked in the chosen unit, going through the undocking motion (opening the arm of the docking unit and moving backwards for several seconds in order to release the aMussel). Blue segments denote that no phase is currently active and the aPad is waiting to receive data from the aMussel or is in a timeout/idle state. The magenta X marks the GPS location the aMussel last reported.

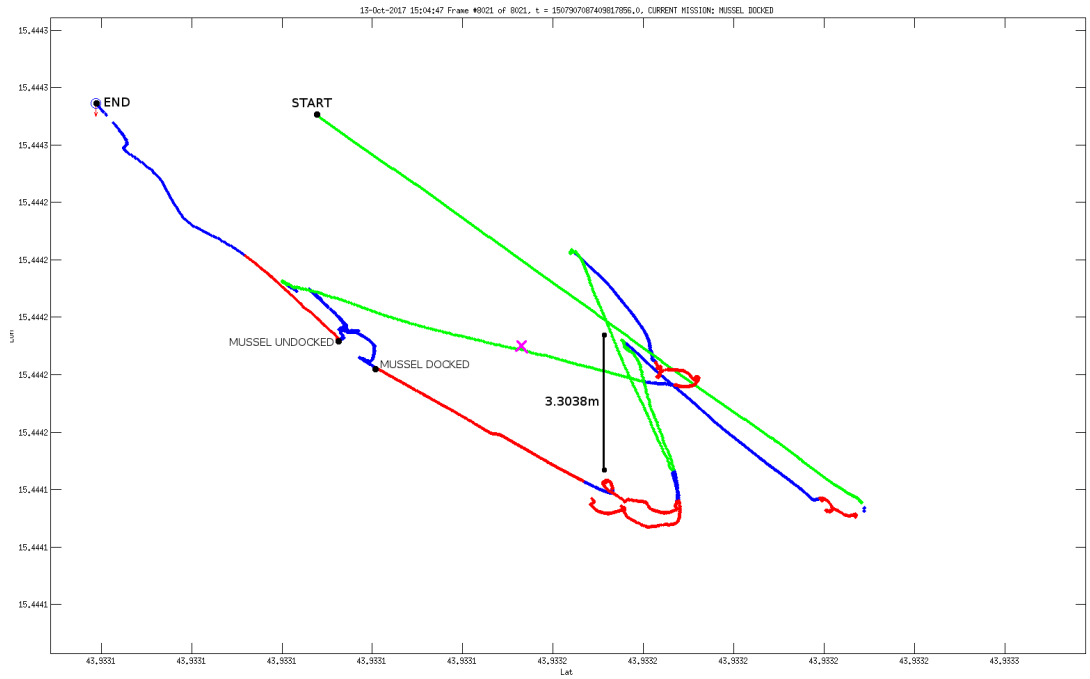


Figure 2.25: Full run of one docking mission with phases and key points shown.

A video showing the replay of processed data of the same search, docking, and undocking mission can be found at the following link: `https://www.youtube.com/watch?v=LCPmYnDt7z8`.

A factor which assists with increasing the probability of convergence of the docking procedure is the fact that the mussel's GPS fix improves over time. Rough analysis of aMussel GPS data showed that after an interval of approximately 10 minutes measurements displayed a significant improvement, with errors reducing from an initial 20+ metre offset to under 10, allowing for far more successful docking attempts by the aPad.

Analysis of the visual servoing attempts shows that, while a great degree of tolerance to different lighting and weather conditions has been achieved, the glare caused by attempting docking while directly facing the sun is still an issue for the image processing algorithm. This is helped by the aPad attempting approach from different sides of the aMussel.

2.6.3 Structured docking experiment

The structured docking experiment took place at lake Jarun in Zagreb in June 2019, with the goal of validating the final developed system. The experiment location and a shot of the experiment in progress can be seen in Figure 2.26.

Figure 2.26: Structured docking experiment at lake Jarun. Experimental environment (left) and experiment in progress (right).

In order to test the docking algorithm in a controlled and verifiable setup, an aMussel was anchored using rope and metal weights and its position was measured using the aPad's high-precision GPS. This measurement was then used as a ground truth reference for aMussel position throughout the experiment and provided information about position offsets and distances of the aPad from its goal.

After setting up the anchored aMussel, four primary aPad starting positions were chosen along a circle with a radius corresponding to an empirically established maximum distance of consistently reliable docking. This maximum distance was determined to be between 4 and 5 metres of distance from the aMussel - note that docking from further away is possible (with "lucky" cases of successes from as much as 15 metres away), but is not necessarily consistent and thus not considered reliable or suitable for an autonomous system. The starting positions were then distributed along the circle as seen in Figure 2.27 so that the four major approach directions were included, in order for the effects of sunlight and glare to be accounted for during the experiment. The starting point to anchor point distances reported by Neptus were, starting form the northernmost point and continuing clockwise: 4.18m, 5.11m, 4.13m, and 4.16m. Note that during the final run of the experiment presented here, it was run twice from the direction of the first starting point and thus five docking attempts are shown.

The aPad would be manually sent to each of the marked starting points and turned so the aMussel was not immediately in view, after which the docking algorithm would be started. After each successful docking, the aMussel would be released and the aPad sent to the next point. The experiment was concluded once a continuous run of docking attempts from all directions was achieved.

Figure 2.27: Starting positions for docking attempts around an anchored aMussel as envisioned (left) and marked as goalpoints in the Neptus C4I Framework used for aPad mission supervision and control (right) [82].

Figure 2.28 presents the trajectory of the aPad during the experiment, with denoted colour-coded segments during which each of the mission primitives (docking/undocking, going to next point, and no mission selected) were active. The blue circle marker represents the position of the anchored aMussel, while the aPad's heading is denoted by a red arrow.

During the experiment, horizontal, vertical, and total offset data was recorded. The horizontal and vertical offsets refer to the position of the aMussel relative to the center of the image frame, as output by the EKF. The aMussel total offset refers to the calculated distance between the current position of the aPad and the saved position of the anchored aMussel. These are all shown in Figure 2.29, where the five docking attempts are all evident: the distance from the aMussel decreases, and it simultaneously moves towards the bottom of the image while being kept horizontally centred.

Note that for about three minutes between the fourth and fifth docking attempt the aPad was not actively performing the docking experiment, but rather having minor maintenance performed and entangled lily pads and debris removed from its hull and docks. This data was left included for consistency as it was recorded during the experiment, but it has been greyed out in plots as it is not relevant or informative.

During the fifth docking attempt shown (starting at 510s on the graphs in Figure 2.29) it can be seen how the aPad briefly loses sight of the aMussel, returning to the Search state, then finds it once again after some rotation, continuing and concluding the docking attempt successfully.

The anchoring of the aMussel and the aPad's position estimate are both imperfect, and continued docking attempts nudged the anchor slightly from its original noted position,

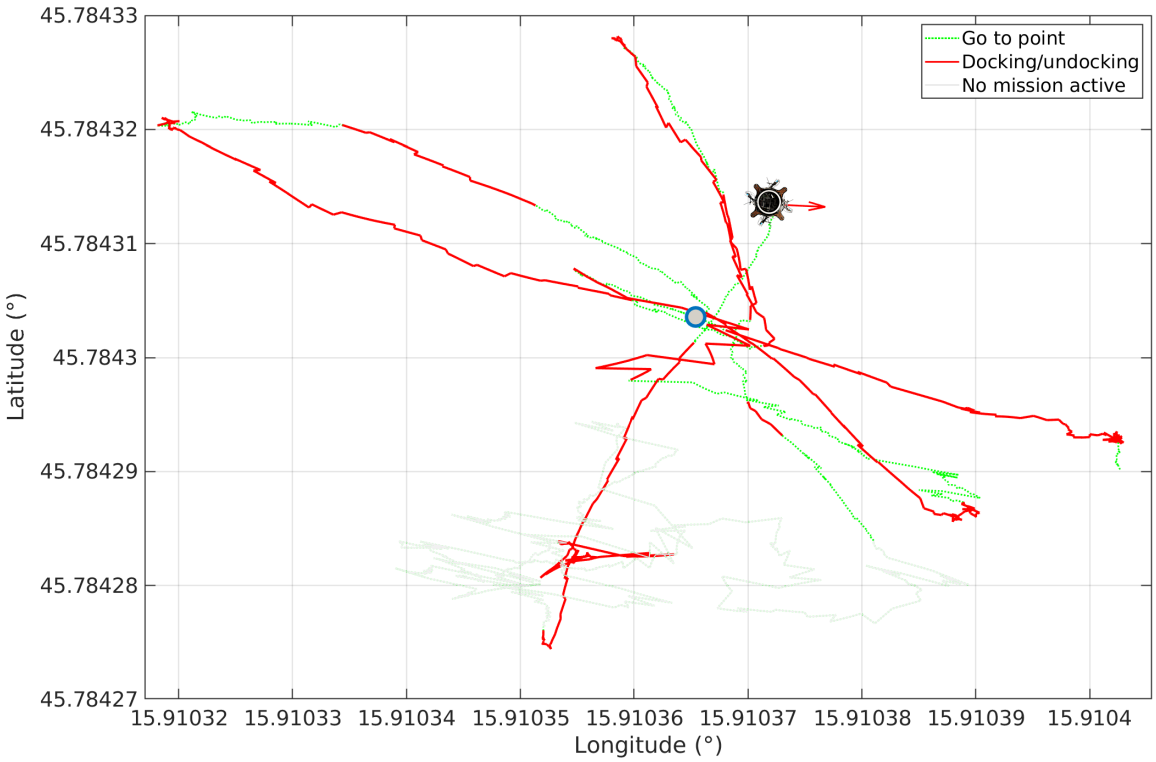


Figure 2.28: Structured docking experiment full mission trajectory. The blue circle marker represents the position of the anchored aMussel.

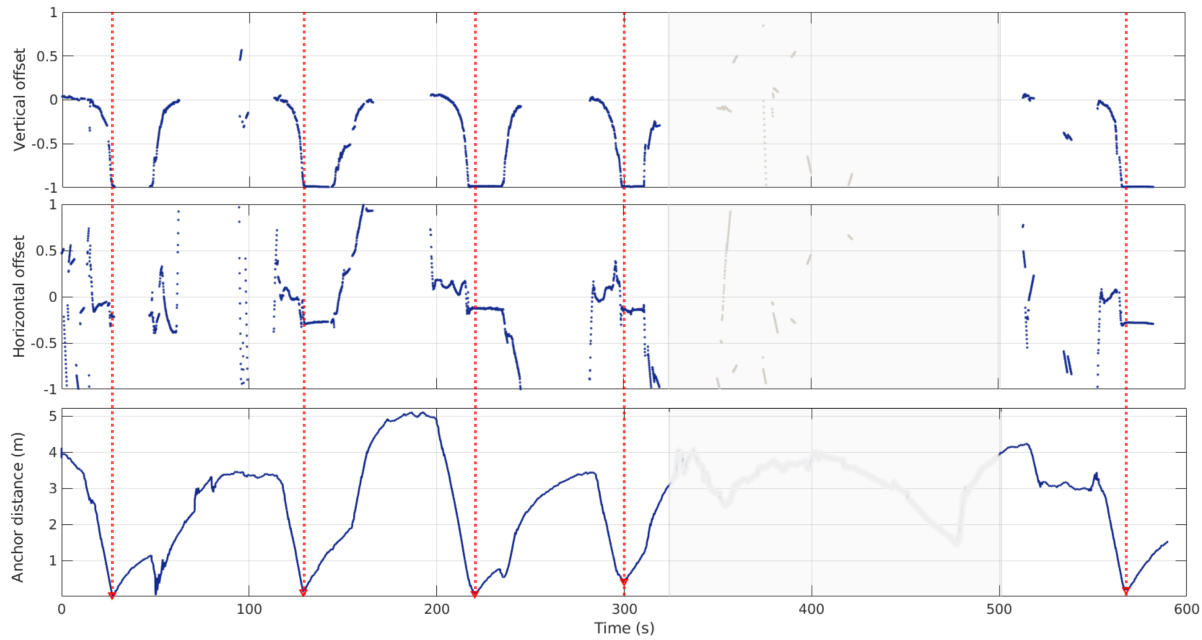


Figure 2.29: Offsets during the docking experiment. Five separate successful docking attempt completions marked in red.

so the distance between the aPad and the aMussel can be seen approaching zero but still having a slight offset. Due to the length of the rope used for anchoring, the aPad also

had some leeway to move with the aMussel while holding it docked, but upon release the aMussel moved back towards its original position - hence the small rise and fall in distance just after each docking concluded.

The jagged quality of the lines representing the trajectory of the aPad during the experiment and the small jumps in position are a result of the aPad's navigation filter updating with GPS measurement corrections received in fixed time intervals.

The previously described implemented filter heuristics turned out to be very helpful in dealing with brief flashes of false positive detections, eliminating problematic reaction to them all but entirely. The aPad was able to successfully dock the aMussel from each direction, and was able to resume and successfully conclude a docking attempt after losing sight of its target.

2.6.4 Challenging environment test

The realistic environment docking experiment took place during a prolonged set of field trials for the subCULTron project in Venice in July 2019, on a particularly windy day with rough, choppy waves making for challenging conditions but providing a good opportunity to test the operational limits of the docking system (see Figure 2.30).

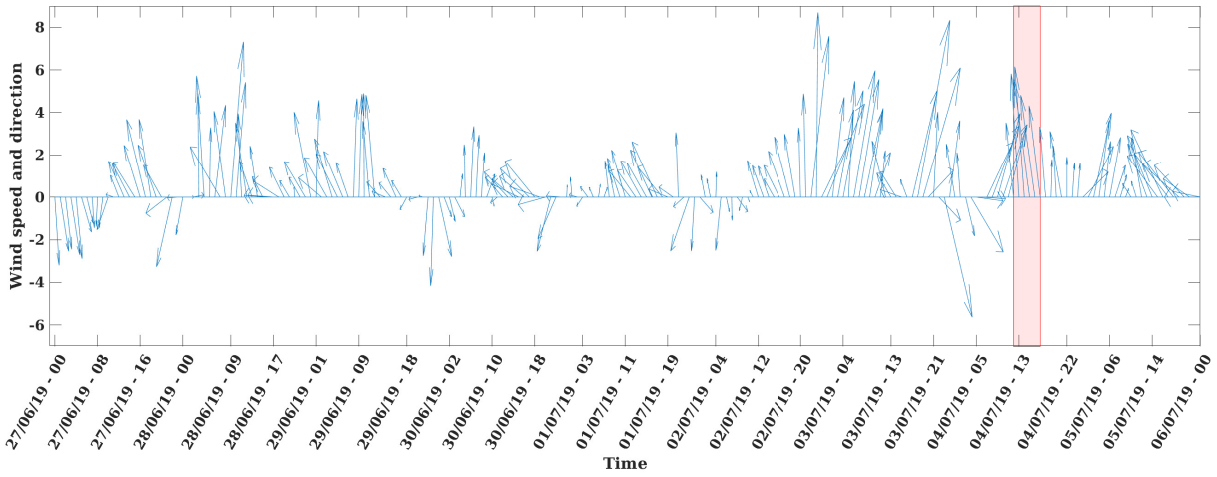


Figure 2.30: Wind speed and direction during the 2019 field trials, as collected by the Malamocco weather station. The docking experiment taking place during July 4$^{\text{th}}$ is shown in red. Note all other wind peaks occurred during night, with a storm taking place the evening of July 3$^{\text{rd}}$.

The subCULTron robots were deployed off a pontoon, with a designated open sea workspace next to a relatively low-traffic water route.

A video including visual feeds from the Kinect, image processing and EKF data, and a mission replay constructed from the aPad mission data logs for the autonomous docking field experiment can be seen at at the following link: `https://www.youtube.com/watch?v=Agky3vv6Mh4`.

Figure 2.31: On-site experiments in the Venice lagoon. The pontoon off which robots were deployed (left) and field trials in progress (right).

Unlike the structured experiment, no data about the aMussel's position was available. Image-derived data and information about when each of the docking attempts started and concluded were collected.

Figure 2.32 presents the full trajectory of the aPad during the experiment, with colour-coded segments showing when each of the mission primitives were active and markers showing the moment of each docking and undocking. The aPad's heading is denoted by a red arrow. The X markers designate successful docking attempts, while the black diamond markers signify undockings. The aMussel was drifting in currents of varying strengths towards the northwest of the given image, leading to noticeable differences in position between where the aPad leaves it after undocking and where it is later caught (i.e. the black markers on the trajectory plot).

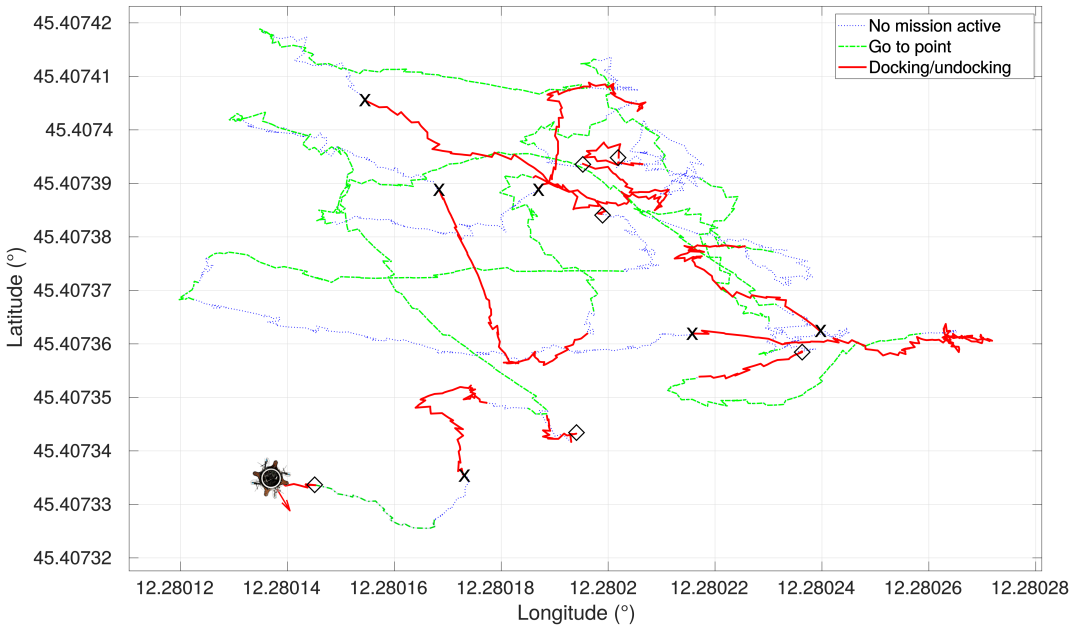


Figure 2.32: Full mission trajectory of the Venice docking experiment in challenging conditions.

Figure 2.33 gives a timeline of the image offsets detected by the hue thresholding and

the neural network, as well as the final filter output for a sequence of six successful docking and undocking attempts. Both docking attempts and undockings can be clearly seen, as the offsets reflect how the aPad approaches the target and captures it, then, after holding it for a while, releases it and moves away.

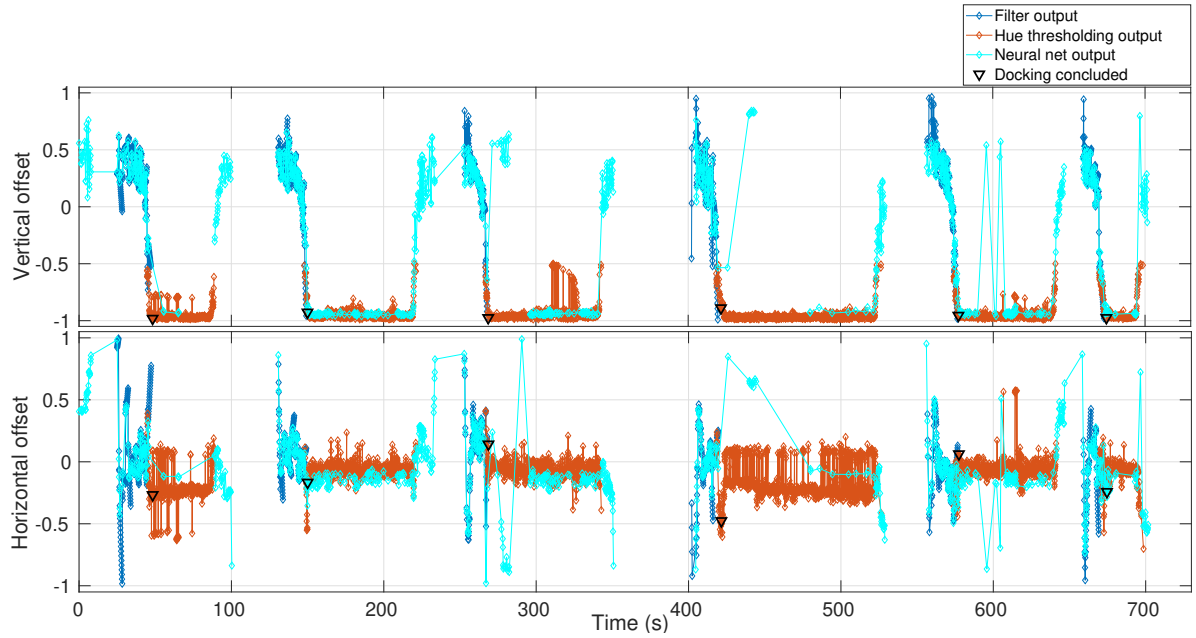


Figure 2.33: aMussel image offsets during the docking experiment in the field. Triangular markers denote where each separate docking attempt ended.

As noted in Section 2.5.1, experiments in real conditions in the lagoon led to a tuned increase in all regulator parameters, as the ones previously used made for an aMussel-catching approach that was very slow and tentative in the presence of waves and wind and sea currents. To accommodate the aMussel bobbing up and down on the waves and briefly being lost from the aPad's view, the parameter for the tracking filter heuristic that ensures continuous measurements and rejects fleeting outlier detections was set to only require 3 consecutive frames of detection as opposed to the original 6.

The experiment showed that the mechanical design of the docking system with vertical offset tolerance was sound, as even with waves causing considerable displacement, the aMussel was successfully caught. The plexiglass-encased vision system also proved robust to these conditions, working even with the casing being sprayed and splashed with water as can be seen in Figure 2.34. Note the neural network output being a frame behind the hue thresholding, and both the result images demonstrating ROI cropping.

The algorithm and implementation proposed for the autonomous docking and charging have proven to fit their intended use in the subCULTron heterogeneous robotic swarm quite well, enabling surface platforms to dock smaller floating sensor nodes using a visual servoing approach, successfully starting wireless energy transfer in a variety of testing conditions with very simple changes in parameter tuning. The mechanical design of the

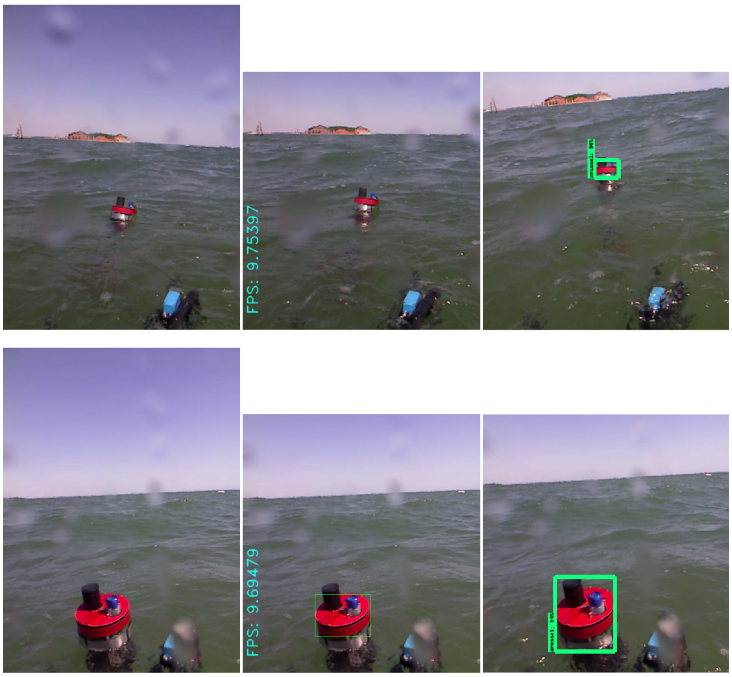


Figure 2.34: Two examples of image processing on the aPad during docking experiments in a realistic environment. From left to right: original image, hue thresholding output, neural network detection output.

aPad's docking unit and aMussel top cap ensured appropriate transmitter and receiver coil alignment and proximity, and thus a consistent and satisfactory efficiency during energy transfer. The developed system has regularly been used to charge aMussels during field trials.

The autonomous docking and wireless charging capabilities of the swarm represent a beneficial cooperative interaction among agents improving the functioning of the swarm as a whole, and were also a prerequisite for the implementation of various decision-making algorithms aimed at scheduling aMussel pickups by aPads and examining the cumulative effects of energy transfer systems on the longevity of the subCULTron swarm, in the interest of maximising it and ensuring satisfactory environmental monitoring.

Chapter 3

Multi-robot task assignment and low-level heuristics

3.1 Introduction

This chapter presents an overview of the proposed layered structure of the decision-making algorithms in the system, as well as a more detailed examination of the problem scenario that is being studied. Chosen methods of initial division and assignment of aMussels to aPads are described - primarily differential evolution and k-means clustering. A description is given of the collection of methods belonging to the bottom layer of the proposed hyper-heuristic decision-making and task allocation approach - the situational low-level heuristics, or the methods making up the heuristic selection pool. Performance indices meant for scoring the performance of individual heuristics and evaluating various swarm state parameters are discussed. Finally, some conclusions are drawn from results achieved in a simulated marine environment and early experiments performed on real vehicles.

For the purposes of devising low-level heuristics, the aPad task allocation problem can be described as a type of vehicle routing problem. Heuristic solution methods targeting specific variants of the vehicle routing problem such as the vehicle routing problem with time windows, the capacitated vehicle routing problem, the multi-depot vehicle routing problem, or the vehicle routing problem with stochastic demand have been studied extensively, with each heuristic being designed, implemented, and fine-tuned to fit one particular problem type [83], [84]. Since problem characteristics can vary considerably, it may not always be entirely obvious which method will yield the best solution for a particular instance of the problem [85] [86]. Here the aim is to devise several context-specific heuristics with good situational behaviour, then select between them and combine approaches as appropriate.

The decision-making process includes a step of separating aMussels into clusters cor-

responding to parameters such as real-world signal strength or viable travel distance, then assigning aPads to individual clusters as a form of "region of interest" in order to ensure both communication and charging coverage. Clustering is a problem present in data mining, database systems, data compression, and machine learning. It involves partitioning a set of observations into clusters such that the intra-cluster observations are as similar (or close, in the chosen metric) as possible and the inter-cluster observations as dissimilar (or distant) as possible. The other objective of clustering is to reduce the complexity of the data by replacing a group of observations with a single representative observation, leading to easier and faster computation and analysis [87], [88].

Partitioning-based clustering algorithms such as the widely-used k-means algorithm are a specific subtype which organise objects into some number of partitions, where each partition represents a single cluster. The clusters are formed based on a distance function, leading to the formation of only spherical clusters and allowing the clustering results to be influenced by noise. If the specific coverage and charging sequence task assignment problem is being considered, this issue is not particularly problematic, as convex and spherical clusters are indeed the desired result. The k-means algorithm also requires the number of clusters to be known and specified in advance, but can otherwise be completely unsupervised [89] or, in the case of incorporation of some prior knowledge of outcome measures, semi-supervised [90]. Enforcing constraints in partitioning-based clustering algorithms is introducing a form of prior knowledge and has been studied [91], [92]. These constraints frequently take the form of "must-link" and "cannot-link" clauses, though more complex combinations and formulations can be used [93], [94], [95].

Differential evolution is another approach to task assignment and decision making within the swarm. It is a population-based iterative heuristic originally proposed in [96] for global optimization over continuous spaces, later adapted for use in discrete spaces and on sequencing, permutation, and scheduling optimisation problems [97], [98], including the vehicle routing problem [99], [100]. By introducing specific population and gene encoding and devising suitable costs, penalties, and stopping criteria, it is possible to represent a variety of constraints in the solution space [101].

3.2 Problem scenario and decision-making system structure

The decision-making scenario studied in this thesis is a representation of the main use-case of the subCULTron swarm - a long-term environmental monitoring mission. The studied problem involves a number of aMussels deployed and collecting data from their environment and broadcasting their sensor measurements, and, should they find themselves in

need of battery charging, charging requests including their estimated global position and battery status. One or several aPads are in the vicinity to receive these broadcasts via Wi-Fi (in the rare cases the aMussels are on the surface) or acoustic communication (if the aMussels remain on the seabed). The aPads communicate with each other via Wi-Fi and keep track of their own global position, as well as the position of all other aPads within communication distance - thus, all relative distances and eventual movement costs are considered known to all aPad agents in the system. Extra checks and communication requests before starting any task planning exist so the agents are working with the latest and most relevant information available. The aPads divide aMussels amongst themselves, collect charging requests from them, and run one from a set of proposed task allocation and planning algorithms before moving to and collecting (via autonomous docking) the chosen sequence of aMussels, while communicating with aMussels and letting them know when it is "safe" to surface for charging, as an aPad is ready nearby.

A graphical outline of the proposed decision-making scenario is shown in Figure 3.1.

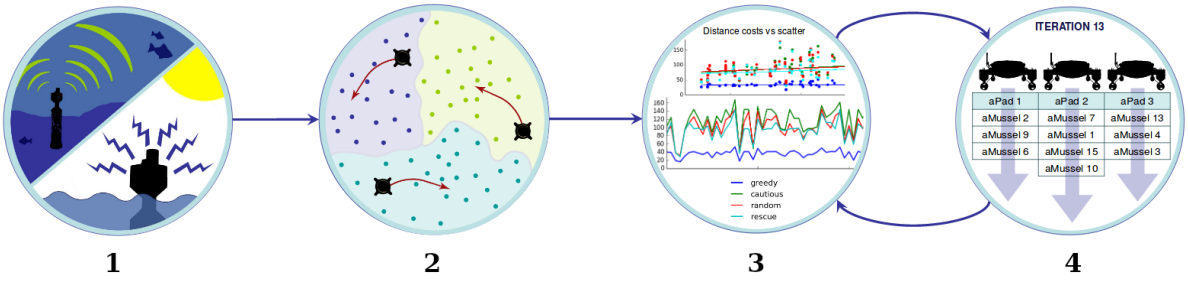


Figure 3.1: Concept of the decision-making scenario within a long-term environmental monitoring mission.

The main steps the scenario is broken into are as follows:

1. A large number of aMussels performs environmental monitoring, communicating via acoustics or Wi-Fi.
2. aPads keep track of aMussel positions and states, splitting the swarm into “patrol areas” by collective decision, depending on the dispersion and required and available energy of the agents.
3. Each aPad evaluates available approaches for charging its own set of mussels using a hyper-heuristic algorithm which selects and scores scheduling methods from a low-level heuristic pool. A record of performance quality scores of each method used in the past is kept.
4. Selected methods create a schedule for collecting up to four mussels. The aPads begin their individual missions and inform any aMussels still at the seabed when they are required to emerge for docking. After all collected aMussels have been charged and redeployed, the aPads return to step 3.

The task allocation system employed in steps 3 and 4 consists of two main decision-making layers and one control layer. The proposed system structure is shown in Figure 3.2. The bottommost layer represents control structures present on the aPad vehicle, many of which are detailed in ([59]). The lower decision-making layer represents the pool of solution-focused and situation-specific heuristic approaches developed to solve the energy exchange-related aPad task allocation problem. The second and higher-level layer contains the hyper-heuristic, an algorithm that selects and switches between the heuristics implemented in the lower layer as the situation demands, with the end goal of autonomous and unsupervised mission planning while optimising energy consumption and distribution. This topmost layer is described in detail in Chapter 4.

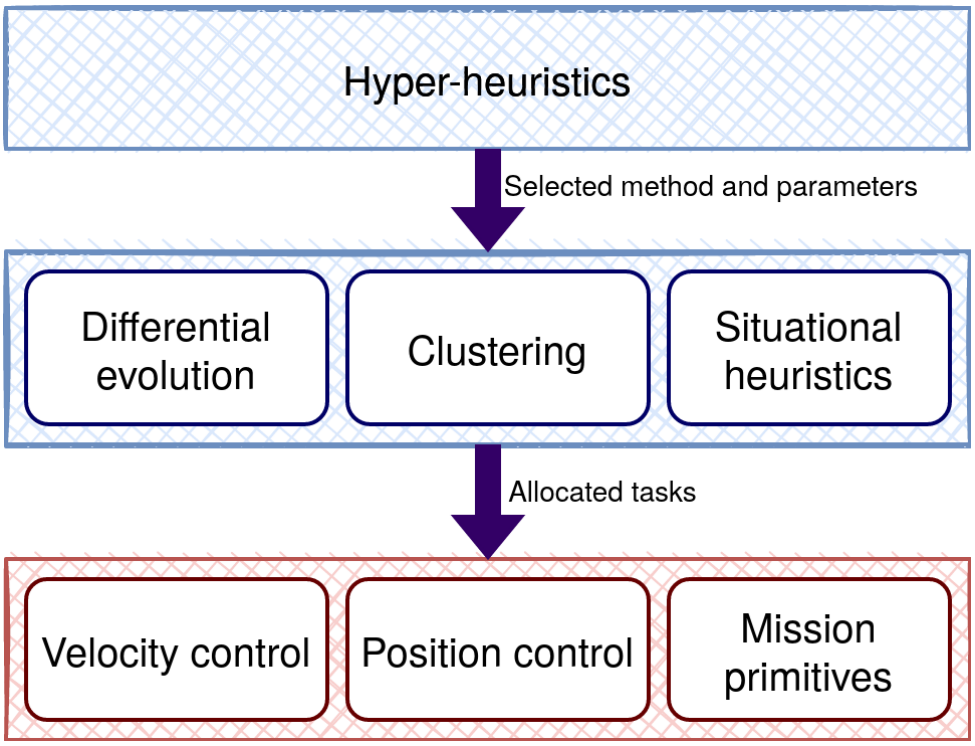


Figure 3.2: Proposed layers of control and decision-making algorithms in the system.

Realisation of the lower decision-making layer includes developing a set of task assignment and sequencing algorithms which use or combine different machine learning paradigms or use them in different ways in order to devise a movement and task/action plan for the available vehicles, primarily concerning charging or moving other robots within the swarm. This includes the clustering method, the differential evolution method, and a variety of situational heuristics. Special attention needs to be paid to outlier or edge cases of particular methods performing very well in certain cases and very badly in others (including potentially not converging upon a valid solution at all), for instance behaving differently in the event of a vehicle failure type disturbance, or needing to deal

with outliers in the physical distribution of the swarm, as this will be a valuable future indicator of performance for the hyper-heuristic part of the implemented decision-making algorithms.

Parameters of the system include (but are not limited to) the number of agents active and present, spatial distribution of agents within the swarm, battery levels of all robots, wind, water current and other sea state data, and aPad thruster health. The relative distances of agents (aMussel to aMussel; aPad to aPad and aMussel) are stored in matrices on each aPad and are regularly updated to reflect the latest known state of the swarm. Thruster failure is a possibility, and each aPad has a so-called thruster health matrix indicating the status of all four of its thrusters which is taken into account during task allocation.

The effects of distributed implementation and working in parallel to exploit the strengths of working in a swarm are interesting to observe. Each aPad can independently run any part of the algorithm, possibly leading to interesting diversity or deliberate specialisation of aPads into certain roles. On the lower level, various approaches to achieving consensus about the solution and achieving convergence can be tested - which is necessary since the algorithm is run on all aPads at the same time, based on the same collected information, but each aPad executes its own search and shares very little information during the process in the interest of minimising communication overhead. Once the best valid solution (in the majority of cases the lowest overall energy cost converged upon by the individual aPads) has been found it is given to all the aPads as the final task allocation they should begin performing.

As all active aPads run the decision-making algorithms individually, once method selection has been completed, and once at least one solution in the shape of aPad task allocation has been reached, there are three main options to consider when it comes to the selection of a single final solution to begin executing:

1. interrupt the search on all aPads as soon as one of them has converged upon a valid solution, then distribute this solution to all agents and begin the allocated tasks
2. wait until all aPads have converged upon a solution, interrupting the wait only after a set longer timeout period in case some fail to converge, then find the best solution among those presented
3. wait for a set timeout period after the first reported solution, and use the best solution that is reported within that time

For the final implementation in this thesis, option 3 was chosen, as it presents a good compromise between ensuring timely convergence for the initial decision, while still allowing for delayed solution improvement as a contribution from another agent in the swarm (and the increased possibility of avoiding potential local minima via diversifying

the search outcomes).

As discussed in Section 2.3, an aMussel draws the most power when it is using its buoyancy system motors, leading to the assumption that surfacing and going down needs to be kept to a minimum during operation. Surface time is undesirable in general as not only does it mean the agent is considered inactive, but while freely floating there is also a risk of the mussels drifting away from desired positions or the experimental area in general, leading to loss of equipment which needs to be avoided. Thus, once an aMussel is docked, it will remain charging until it is fully charged - no partial charging sessions are allowed. aMussels that are charging candidates enter the charging selection pool, the upper threshold for which was set to a battery level of 80%. An aPad will also not begin planning a pickup and charging session until there are at least four aMussels in the charging selection pool, in the interest of minimising movement.

After charging, the aMussels must be returned to their erstwhile positions due to the assumption that they were deliberately and purposefully placed there, on points selected in relation to known anomalous points of interest and study. This was done either manually, using a preplanned mission based on historic data, or as a result of one or more exploration algorithms. The aMussels might also differ slightly from one another and have different sensor modules installed, so their spatial distribution must be respected and preserved. Relocation procedures are a separate and important part of exploration paradigms also investigated within the subCULTron project [36].

3.3 aMussel partitioning and assignment

The second step of the described decision-making scenario, assigning "patrol zones" and aMussels to aPads, means effectively moving from a Vehicle Routing Problem (VRP) - analogous to a Travelling Salesman Problem (TSP) -analogous problem after consensus has been achieved. Two primary methods of partitioning were explored.

3.3.1 Differential evolution

Differential Evolution (DE) is used primarily for the purpose of aMussel cluster assignment, but was also the first task sequencing method implemented on the aPads. If an aPad is assigned a very small amount of aMussels, DE can be used to perform task sequencing within the cluster. Here the algorithm criteria is made to correspond to real-world factors such as water movement and robot battery charge levels, with the genes encoding task sequences for specific vehicles.

The solution candidates are encoded as a vector of ordered integers of length

$$len = n_M + n_P - 1 \tag{3.1}$$

where n_M is the number of aMussels and n_P the number of aPads being considered. For one run of the algorithm, each aPad is assigned a route of maximum length 4 (as each aPad can only hold and charge 4 aMussels at a time) or until all aMussels are distributed among the available aPads. The integer zero is used as a delimiter between encoded routes. Available aPads and aMussels are all represented using individual integer IDs. Thus, an example of a valid solution vector s with 3 aPads working to collect 7 aMussels would be $S = [620175304]$, meaning that in this case aPad 1 will move to collect aMussel 6, then aMussel 2; aPad 2 will move to collect aMussel 1, followed by aMussel 7, aMussel 5, and finally aMussel 3; and aPad 3 will collect aMussel 4. This example result of the algorithm is shown in Figure 3.3.

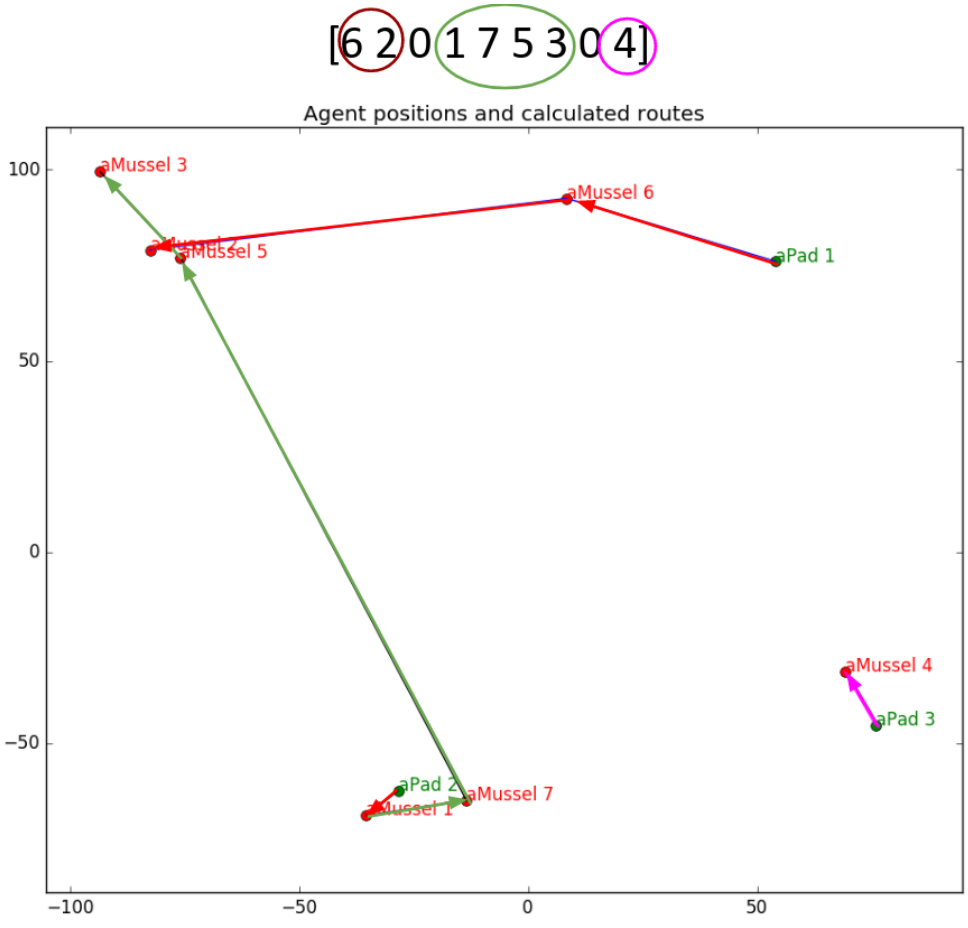


Figure 3.3: Example aPad routing solution using differential evolution.

The cost function for the search algorithm is formulated around energy efficiency, meaning that using up aPad energy to traverse bigger distances is penalised, whereas

transferring more energy to aMussels that need it is rewarded. The energy sum function used is (3.2):

$$f_{cost} = K_{idle}E_{idle} + K_{charge}E_{charge} + K_{move}E_{move}, \quad \begin{matrix} K_{idle} \geq 0, \\ K_{charge} \leq 0, \\ K_{move} \geq 0 \end{matrix} \tag{3.2}$$

Where K_{idle}, K_{charge}, and K_{move} represent weight factors for tuning the optimisation, E_{idle} represents basic idle energy used up by the aPads, E_{charge} is the energy transferred from the aPads to the aMussels (calculated from battery state information), and E_{move} is the energy spent on aPad movement (calculated from distance matrices). Penalties are also in place to ensure only valid routes are generated in the final solution (such as no aMussels being assigned twice, no aPads given tasks that are beyond their battery capacity). Requests for transport without charging enter the algorithm as charging requests with lower priority, since they will provide no contribution to the cost function in the form of transferred energy. For the example used above, the trend of minimisation of aPad movement energy costs (expressed in abstract units that can be correlated with real-life battery voltage/state of charge estimation) is shown in Figure 3.4.

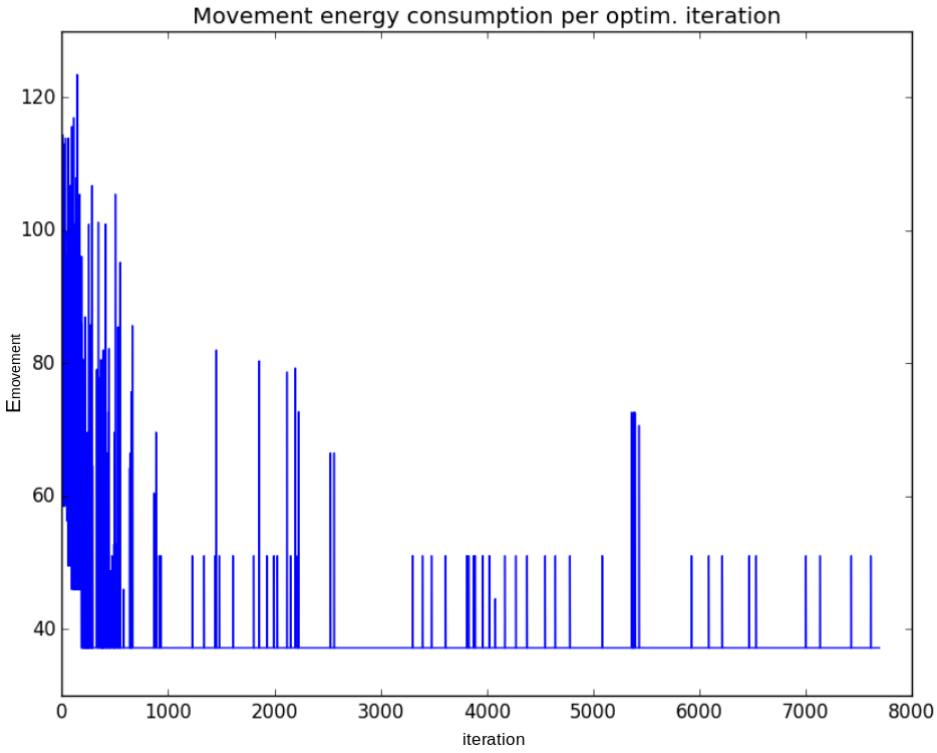


Figure 3.4: aPad movement energy trends over iterations of the differential evolution algorithm. Energy is here given in abstract units.

A significant issue with the differential evolution approach is the length of execution, as well as scaling issues with the solution space vector. As the number of agents increases, convergence upon a valid solution becomes increasingly slow, as well as unlikely to happen at all. Thus, a need for at least one method of partitioning the pool of charging candidates.

3.3.2 Clustering

The method of k-means clustering is used to split the aMussels into groups, and then assign groups to individual aPads. The size of the final aMussel clusters can be modified, leading to several possible implementations and achieving different behaviours. By splitting the map into a number of clusters equal to the number of aPads, each aPad can be assigned a certain "region of interest" - the region where it will operate and charge aMussels. An example of the clustering algorithm is shown in Figure 3.5. It is here assumed that each aPad will simply attend to the cluster whose centroid it is closest to.

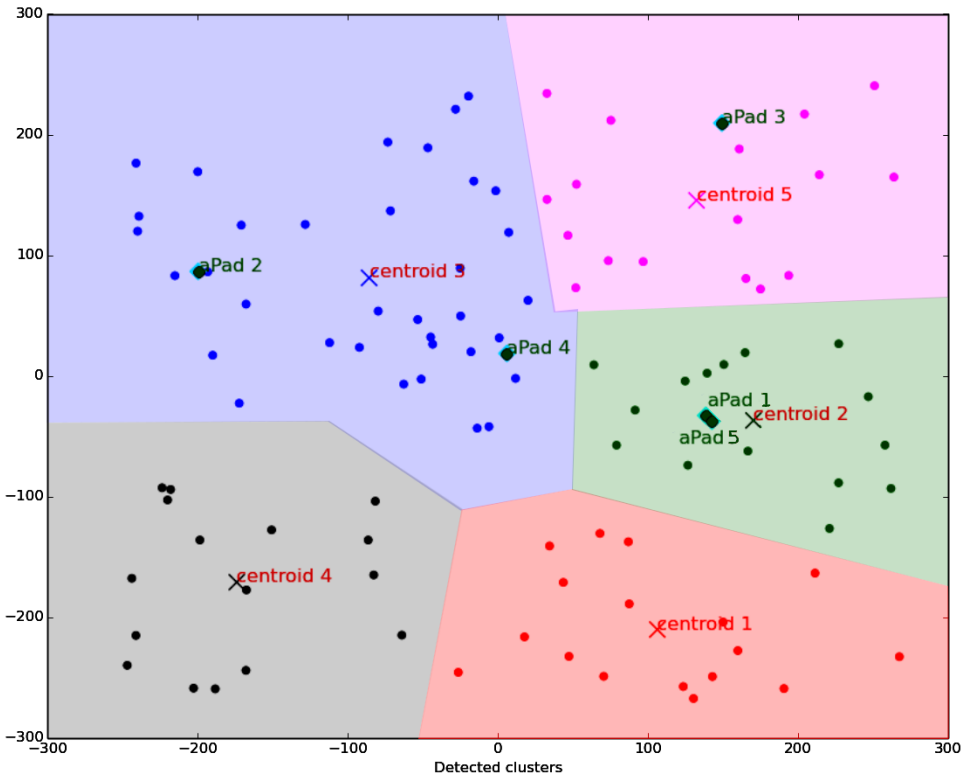


Figure 3.5: Clustering example with 120 aMussels split into 5 clusters, to be assigned to 5 present aPads.

Average clustering execution time was 0.9361 seconds when run on 40000 sets of 120 aMussel positions randomly generated from the same interval (Figure 3.6), clearly demonstrating no execution time or convergence issues like those present with the differential evolution algorithm.

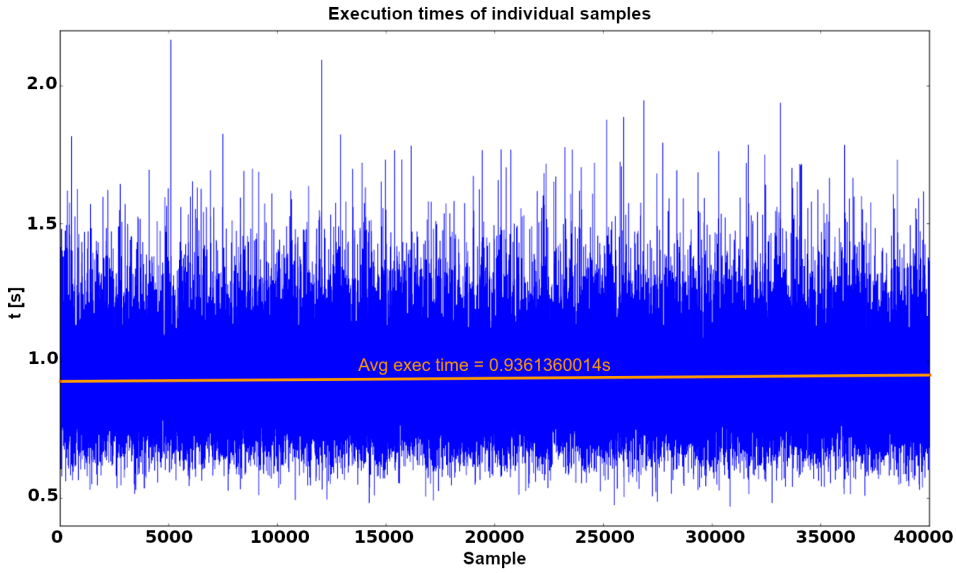


Figure 3.6: Clustering algorithm execution times over a large amount of generated samples.

3.3.3 Combined approach

In a combined approach, the previously described differential evolution optimisation can be used to initially assign aPads to regions by assigning aPads to cluster centroids, then to assign aMussel pickup sequence routes to aPads within each single region. A flowchart of this combined approach can be shown in Figure 3.7. The output cluster regions given by the k-means algorithm serve as limits for evolution algorithm searches, greatly affecting search speed and convergence, and "charging requests" taken into account are the energy demands of the entire aMussel cluster.

For work in real-world conditions, it is necessary to consider the effects of disturbances such as water current and wind. In the current implementation, this is handled by the introduction of a simple 2D vector that affects position and distance matrix calculation (3.3):

$$\vec{v}_{current} = C_x\vec{i} + C_y\vec{j} \tag{3.3}$$

With C_x and C_y defining the strength of each directional component. Should these effects drive the agents in directions that might be considered useful, the cost function will be affected appropriately, and so the movement related to them will be harnessed in a positive sense. An example output of aPad to cluster assignment can be seen in Figure 3.8. Current strength and direction is represented by the overlaid arrow. In the example without current, after clustering, aPad 1 was assigned to cluster 5, aPad 2 was assigned to cluster 1, aPad 3 was assigned to cluster 2, aPad 4 was assigned to cluster 3, and aPad

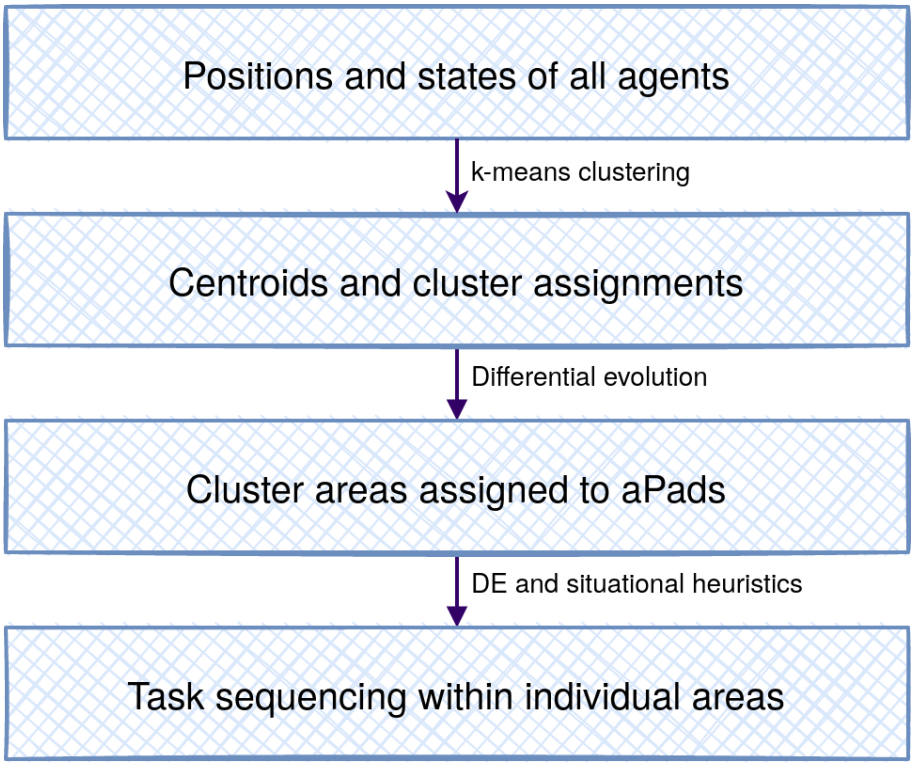


Figure 3.7: Combined clustering and differential evolution approach to aPad task allocation.

5 was assigned to cluster 4. When a current is present, the assignment changes, and now aPad 1 is instead assigned to cluster 2, aPad 2 to cluster 5, and aPad 3 to cluster 1. The total cost function of the aPad assignment has also increased.

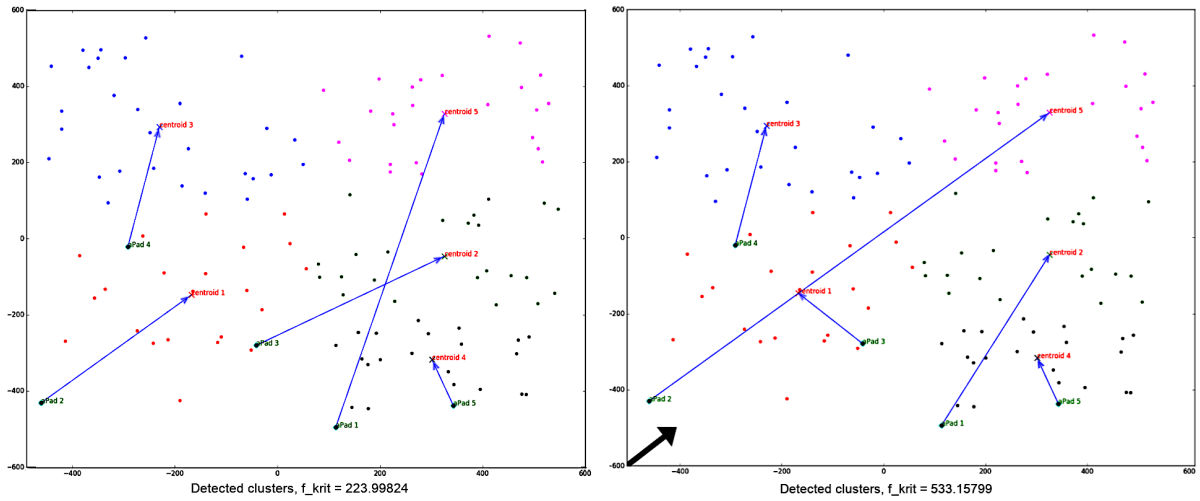


Figure 3.8: Output of combined approach to aPad task allocation, aPads assigned to clusters using differential evolution. Comparison of case with no current (left) and current present (right).

3.4 Performance indices and situational collection/redeployment strategies

As the subCULTron swarm developed and went through testing in a variety of environments, a number of requirements for its behaviours and capabilities emerged. Preventing critically drained aMussel agent batteries from running out and stopping monitoring operation by going into a low-power sleep mode is one of the main goals of the charging system. Since the number of aPads is very small compared to the number of aMussels, rationally using aPad energy becomes crucial, meaning aPad movement should be optimised. Due to the fact most of the aMussel's sensors are on its top cap, an aMussel is inactive while on the surface, meaning that it cannot perform its function while being charged - hence the need to minimise aMussel charging time. aMussels deployed at the edges of the swarm are at higher risk from floating away if, for example, stronger currents or marine vessel-related disturbances manifested at the borders of the testing area, leading to potential robot loss. It is also important that, in a usual monitoring scenario, aMussels have been deliberately deployed over a certain area and at specific measurement locations for a purpose, hence the need to preserve aMussel positions and the limits of the monitored area, keeping the desired measurement and observation points covered by active agents as much as possible.

All of these observations led both to the choice of relevant swarm performance indices, as well as the definition of the situational low-level heuristic methods of task sequencing present in the heuristic selection pool.

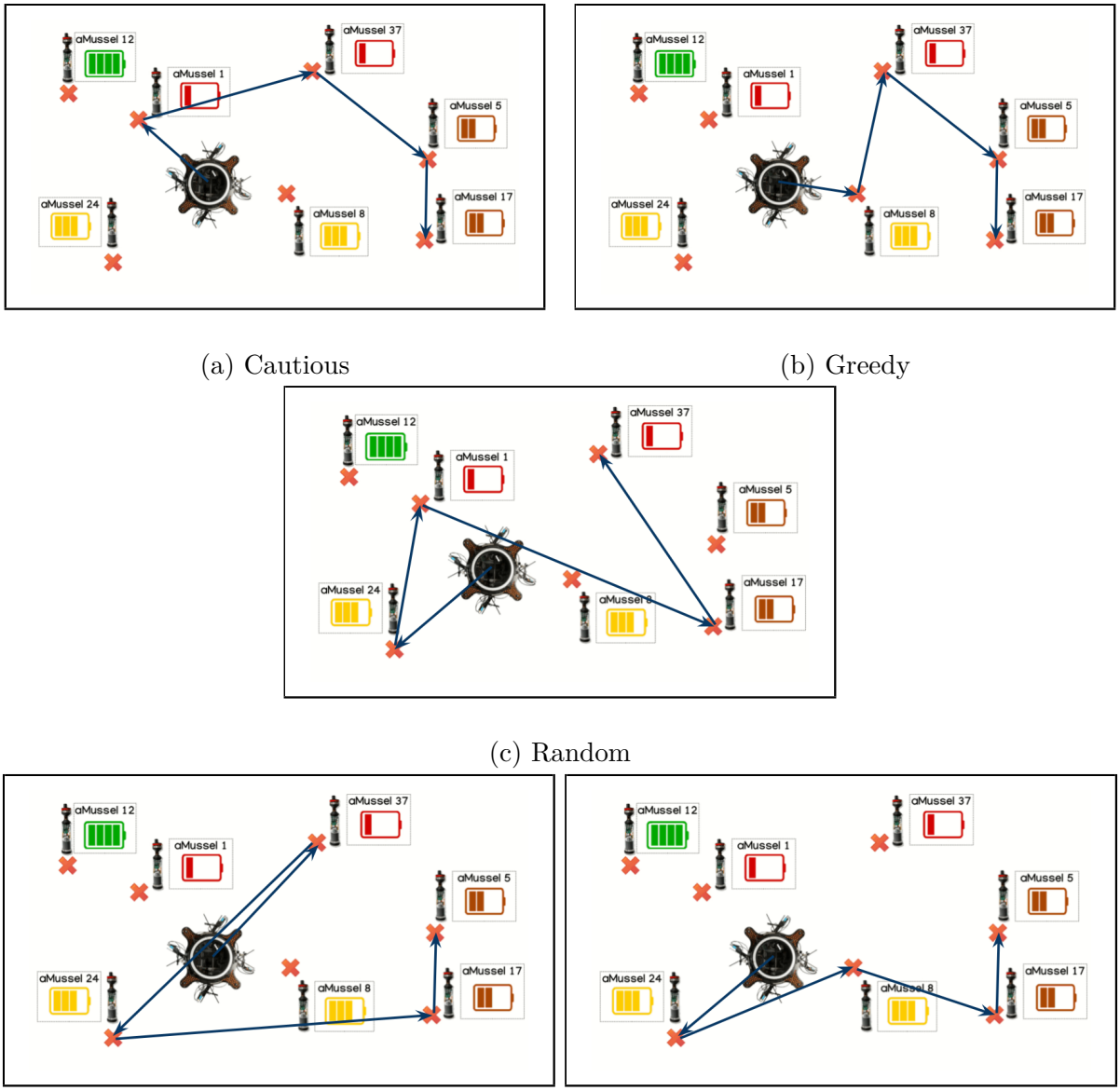
The *Cautious* heuristic does not consider aPad movement at all during planning, instead focusing on collecting the aMussel with the currently lowest battery state in each step, i.e. the aMussel that could be said to be most in need of charging, aiming to prevent complete depletion and potential loss of any agent (Figure 3.9(a)).

Greedy means the aPad moves to collect the closest available aMussel in each planning step. This heuristic aims to reduce aPad movement in a very simple way - there is no guarantee of global movement minimum being achieved, but the planning time required for it is very low (Figure 3.9(b)).

Random represents a purely stochastic element in the system, using a random uniform distribution to select which aMussels in the charging pool to collect (Figure 3.9(c)).

In *Rescue* the aPad collects aMussels furthest away from the cluster centroid, with the idea of preserving area coverage and outliers, while also avoiding local minima caused by selecting for near-optimality with regards to aPad movement (Figure 3.9(d)).

Rush is a method that is uptime-focused, in which the aPad aims to charge in short bursts those aMussels with the highest battery level that are still in the charging pool as



(a) Cautious (b) Greedy

(c) Random

(d) Rescue (e) Rush

Figure 3.9: Examples of aPad trajectories using each of the low-level heuristics from the pool.

candidates, and in doing so aims to “waste" minimal time on charging (Figure 3.9(e)).

After charging of a full load of four aMussels is completed, aMussel redeployment is done using *Greedy Redeploy*. During Greedy Redeploy, the aPad uses dynamic positioning to stay at the position of the last of the four aMussels it has collected for charging, then, once all aMussels are charged, it moves to redeploy them in order of current proximity to their initial positions. This redeploy sequence minimises aPad movement and will be the exact reverse of the sequence generated during Greedy collection planning, whereas for other low-level heuristics this will not be the case.

Due to the nature of the end goal of the swarm being a long-term monitoring mission of slow-changing variables, aMussel uptime, defined as the percentage of total mission time

during which the aMussel was actively performing useful work and capable of collecting relevant data, was chosen as the main performance criterion and benchmark.

As aMussels have most of their sensors mounted on their top caps, they must be fully submerged to properly collect measurements - meaning they are useless for this purpose during charging, and time spent charging is thus not part of the total aMussel uptime. Of course, an aMussel is also not considered active and contributing to its uptime while its battery is depleted below the lower operational threshold. Each aMussel's activity state is logged every time step for the entire duration of the experiment as a simple binary value - it is either active or inactive. Total uptime for each individual aMussel is then calculated as (3.4):

$$T_{up,i}[\%] = 100 \cdot \frac{\sum_{k=1}^{N_{step}} a_{i,k} \Delta t}{t_{max}} = 100 \cdot \frac{\sum_{k=1}^{N_{step}} a_{i,k}}{N_{step}} \tag{3.4}$$

where $a_{i,k} \in \{0, 1\}$ represents activity of aMussel i during time step k, Δt is the time step, t_{max} is total mission duration, and N_{step} is the total number of time steps recorded.

During mission replay and analysis, uptime is calculated for every individual aMussel using the total inactive and active time step counts compared to the total time step count of the mission. For performance scoring during mission execution, cumulative uptime is calculated over each charging cycle, then scaled with the length of the charging interval.

Also calculated are the average, minimum, and maximum of uptime for all aMussels present in the experiment, as well as the balance of the aMussel uptime distribution with regards to individual agents by calculating the standard deviation of uptime. The number of currently active aMussels during each time step is also recorded and considered, as in a case without any charging taking place and for a sufficiently short mission time, aMussels could show "good" (if perhaps front-loaded) uptime as there is no loss of activity due to periods of charging, but this would ultimately leave the area completely unsupervised as all agents deplete their batteries and become inactive, which is unacceptable. A loss of active aMussels in the experimental area represents severe loss of monitoring capability and thus of valuable data.

Charging-based cost is a representation of charging demand from the aMussel side, i.e. an abstracted battery percentage difference from a fully charged state. In the current interpretation, base charging energy cost is always the same no matter which heuristic is used, since the algorithm requires all aMussels within a cluster to be charged.

The other goal to be achieved is minimising aPad energy used for purposes other than charging aMussels, primarily meaning minimising powering its thrusters for movement. The aPad movement cost used by the decision-making algorithms is a simple calculation of Euclidean distance between the current aPad position and the GoTo mission primitive goal

setpoint, meaning the spatial distribution of aMussels (i.e. the scatter of their collective cluster) greatly affects movement costs during a mission. While this useful abstraction can be expected to roughly correspond to aPad battery depletion and a smaller distance to travel will imply less energy expenditure, real world effects and disturbances such as wind and current speeds, as well as the amount of aMussels being carried, affect the vehicle. Vehicle-in-the-loop tests enable the study of these effects, among others. Additionally, in order to reflect the above-mentioned effects, "real" movement cost is tracked, by means of counting every time tick an aPad is moving (1s), effectively representing the total amount of time aPad thrusters were active - a measure of aPad power consumption happening due to movement (3.5):

$$T_{move,i} = \sum_{k=1}^{N_{step}} p_{i,k} \Delta t \tag{3.5}$$

where $p_{i,k} \in \{0, 1\}$ represents whether aPad i is actively moving using its thrusters during time step k derived from its actuating matrix $\boldsymbol{\tau}$, Δt is the time step, and N_{step} is the total number of time steps recorded. If an aPad is moving more slowly for any reason, an increased energy expense will thus be reflected in the score. The aPad's thrust allocation does not change during a mission, meaning that if it is encountering difficulty moving, it will not increase thrust or speed and thus consume more power, but will take a longer time to reach its destination.

A value considered during partitioning and task allocation is the Sum of Squared Error (SSE) of the aMussel cluster - which, as scatter, could be considered an expression of physical distances. All aMussel positions x_i are assigned to c clusters G_j, where $j \in [1, c]$, with centroids $C_j = \overline{x_i}, x_i \in G_j$. SSE for cluster G_j consisting of n aMussels is calculated as (3.6):

$$SSE_j = \sum_{\substack{i=1 \\ x_i \in G_j}}^{n} (\|x_i - C_j\|)^2 \tag{3.6}$$

Coverage-based cost, or the outlier preservation score, serves to evaluate how well the most scattered and distant parts of an aMussel cluster are preserved. It is calculated as the maximum distance between the cluster centroid and a currently active aMussel (3.7):

$$r_k = \max(\|C_j, \mathbf{x}_{j,k}\|) \tag{3.7}$$

where x_j contains all aMussels in cluster G_j active in step k.

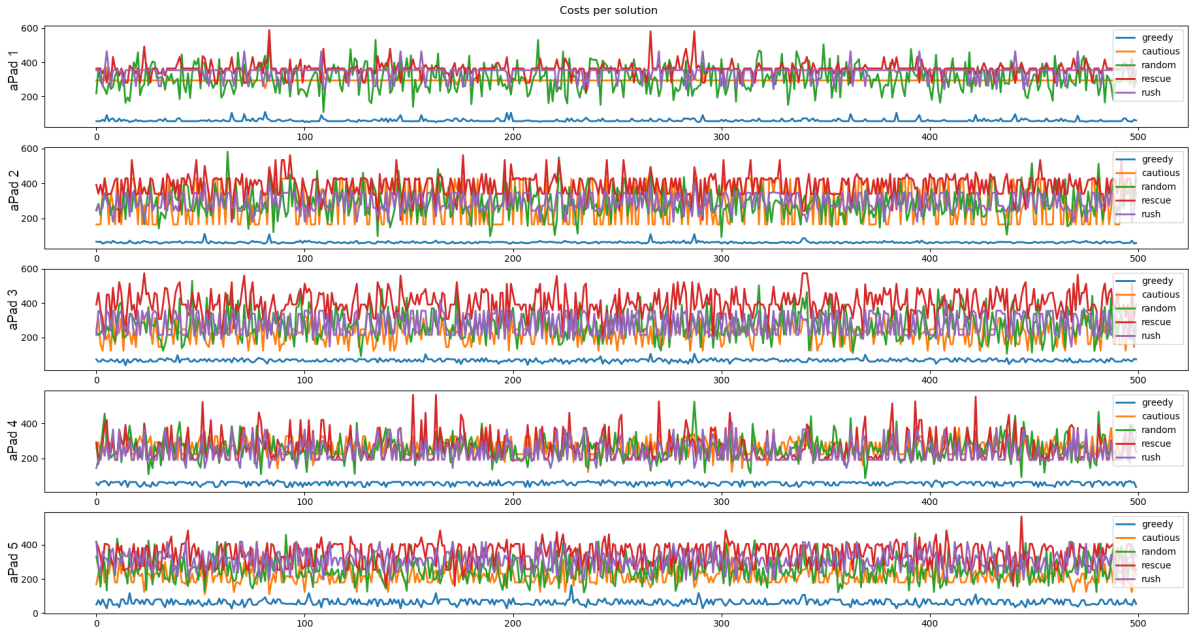


Figure 3.10: Distance cost plot for 500 experiment iterations performed on 5 aPads. Includes 120 aMussels and all 5 low-level heuristics.

3.5 Discrete event simulation

SimPy [102], a process-based discrete-event simulation framework based on standard Python, was used to construct an initial system simulation. The aPads are modelled as shared resources of aMussel processes containing 4 charging stations/docks each. Each aMussel requests docking, and is docked as soon as its turn is up on the scheduled list, and there is a dock available on the aPad it has been assigned to. The charging rate of each aMussel is initially set to one battery "percentage" per unit of time.

The main outline of the simulated experiment was to generate one set of aMussel and aPad locations, then run repeated simulations, repeating initial clustering, scatter calculation, and outputting results of each of the low-level heuristics in order to enable comparison. Running the same heuristic repeatedly (excepting Random) on the same clusters produces the same results as there is no stochastic or local search component.

It is of interest to plot cluster SSE versus costs to see how the values correspond and behave, as it is expected that the scatter measure of a cluster of aMussels will influence the cost of any task allocation solutions applied to it. More aPads lead to more different clusterings and thus a greater variety of SSE scores in the sample of repeated experiments.

A plot of distance-based costs calculated during 500 simulations of each heuristic for 5 aPads and 120 aMussels is shown in Figure 3.10. A comparison of these costs and the SSE values of the generated clusters are shown in Figure 3.11. Charging costs are shown in Figure 3.12.

As expected, in simulations with a number of different generated aMussel clusters,

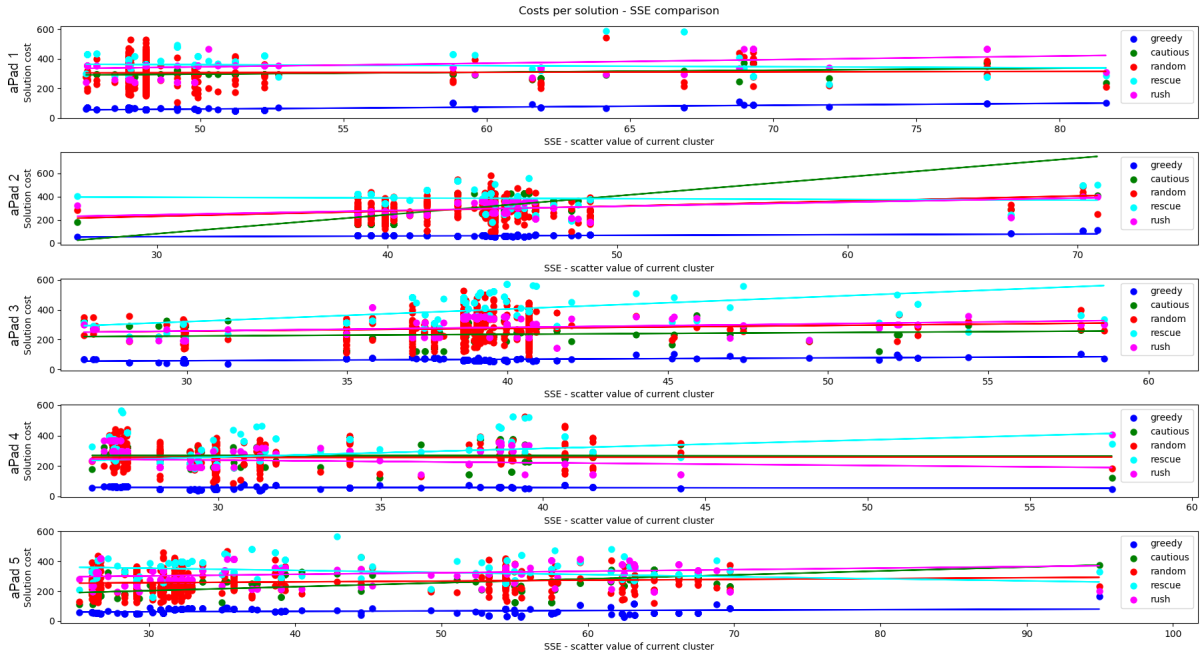


Figure 3.11: Distance cost vs SSE plot for 500 experiment iterations on 5 aPads with 120 aMussels. Simple linear regression is applied to the data.

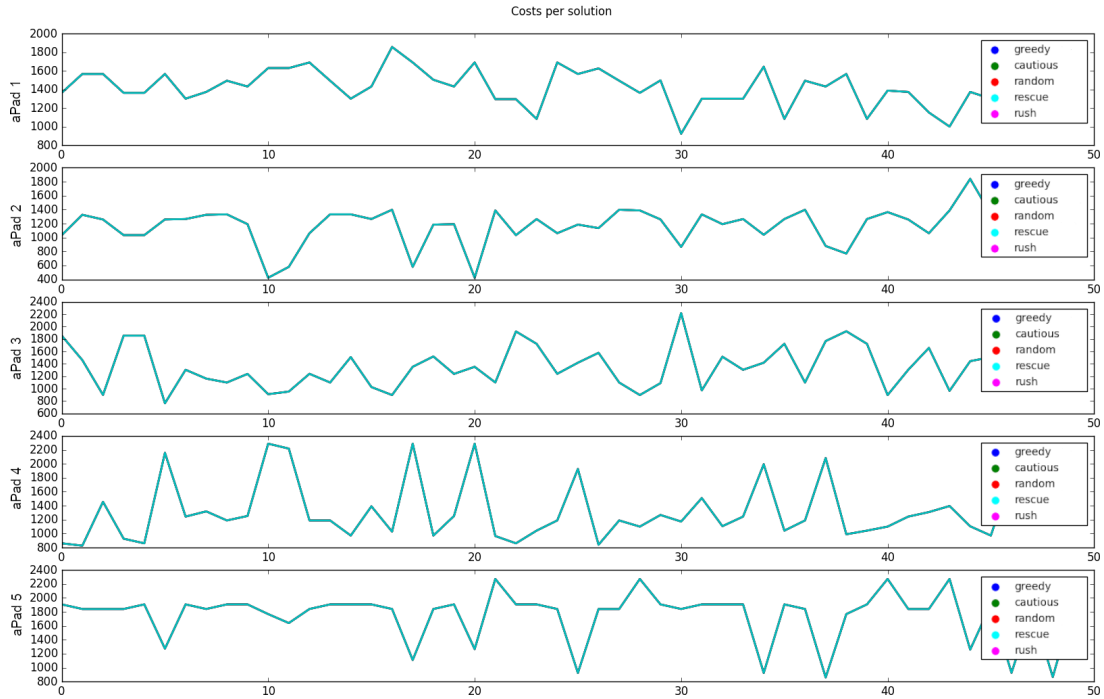


Figure 3.12: Charging cost for all methods - all demand every aMussel be fully charged, meaning charging costs are consistent across all instances.

aPad movement cost was consistently smaller for the Greedy method as opposed to the Cautious method. Noticeably, the Cautious approach solution cost seems to vary more based on SSE than the Greedy approach solution cost (though no strict correspondence seems to be present). Greedy takes distance-related costs into account and minimises

travel, whereas Cautious does not, leading to a more pronounced effect of SSE - which, as scatter, is in fact an expression of physical distances.

One of the roles of the Random heuristic in the system is to provide a baseline, i.e. a successful heuristic "should" be able to perform better than a selection happening completely at random, without any coherent criteria. Distance-cost-wise, it behaves similarly to the Cautious approach, which makes sense, as aMussels and battery levels are initially distributed at random. It is to be expected that this similarity will not be present once a measure of aMussel uptime is introduced.

Each one of its "turns" the Rescue heuristic selects the aMussel furthest from the cluster centroid to be picked up, and, distance-cost-wise, it is by far the worst performing method, with considerable spikes in outlier cases visible in the dataset. The Rush heuristic behaves similarly to the Cautious approach.

The primary, most important and most informative performance index of the complete subCULTron system is aMussel uptime. To simulate this, time needs to be added to all simulations. General time units are used, allowing for flexible simulation and easier application in real-life experimental scenarios.

Travel time was the next addition to the simulation, as a distance-between-agents-based value calculated using a fixed aPad speed of 0.3 distance units divided by time units. This abstraction neglects the impact of currents on aPad speed, as well as the effect of an aPad travelling while carrying aMussels, as opposed to while empty. Travel time from agent with ID i to agent with ID j is thus calculated as (3.8):

$$t_{travel,i,j} = \frac{d_{i,j}}{0.3} \tag{3.8}$$

Where $d_{i,j}$ is the distance from agent i to agent j. Travel to the next scheduled aMussel was in its initial implementation a simple blocking delay, meaning charging processes could not be updated during it, i.e. charging could not take place during it (an acceptable approximation to start with, as travel time is in reality very short compared to charging time). Revised and with travel capability modelled as a separate resource of its own type, the aPad travels to an aMussel while charging others that it has already docked, and is so prepared to collect it as soon as one of the docked aMussels is fully charged and undocked. In this implementation, charged aMussels will be undocked in various places, sometimes where they were picked up (if no mussels are to be docked after them), sometimes near where another mussel was picked up.

aMussel battery charging and depletion rates are variables that can be set separately. An aPad has 4 docking stations to charge 4 aMussels in parallel (docks - a resource with capacity 4), however, it can only be physically travelling to one aMussel at a time (travel ability - resource with capacity 1). aMussels have to request one of the docking stations

and the aPad has to reach them - the travel process is t_{travel} long.

Once docked, the charging processes can be started, and the aPad is possibly travelling in the meantime. The charging process takes an amount of time calculated based on aMussel battery levels at the moment of docking (3.9):

$$t_{charging,i} = (100 - bat_i) \cdot r_{charge} \tag{3.9}$$

Where bat_i is the current battery state of the aMussel with ID i, and r_{charge} is the current rate at which the aMussel battery gains charge.

During the length of the simulation, all aMussel battery levels and charging states are logged (Figure 3.13). Charging state is 0 when not charging, 1 when charging - scaled to [0, 100] for data display purposes. This data will factor into the uptime calculation as, again, aMussels cannot count as active while they are being charged on the surface. The charge/discharge/request pickup/schedule-decision/dock cycle can be repeated. Here the aMussels were set to start at a random depleted battery state from the [20, 40] interval, and only one cycle is shown - the aMussels were left to completely deplete their batteries afterwards.

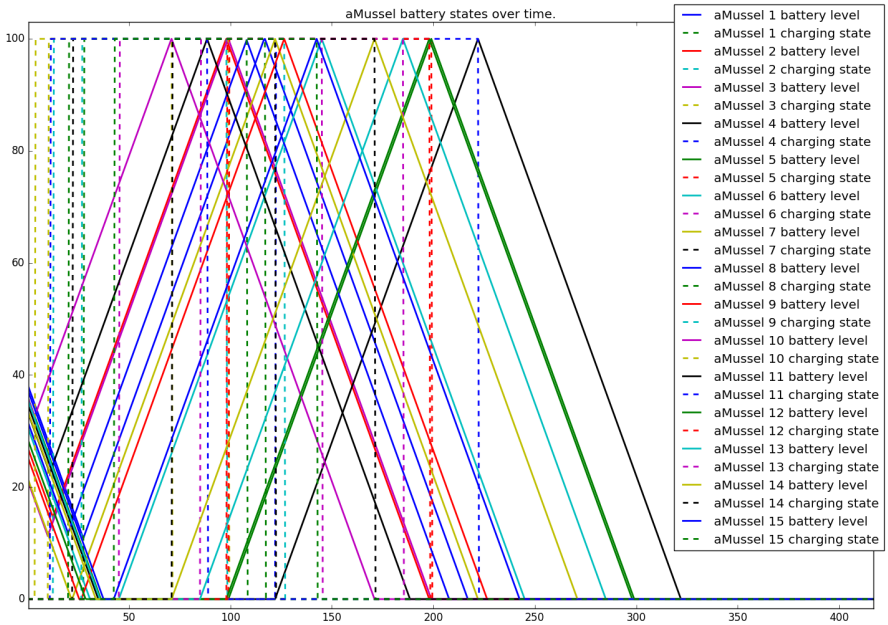


Figure 3.13: 500 time units long simulation of a 3 aPad and 15 aMussel pickup and charging cycle, discharge and charge rate parameters both set to 0.5.

Visualising the states of all aMussels in the system can be difficult to parse and follow (as obvious in Figure 3.13), so two individual aMussels (1 and 3) will be observed separately (Figure 3.14). During the decision making process, aMussel 1 was assigned to

aPad 2 as the second mussel on its pickup route, while aMussel 3 was assigned to aPad 1 as the first mussel on its pickup route.

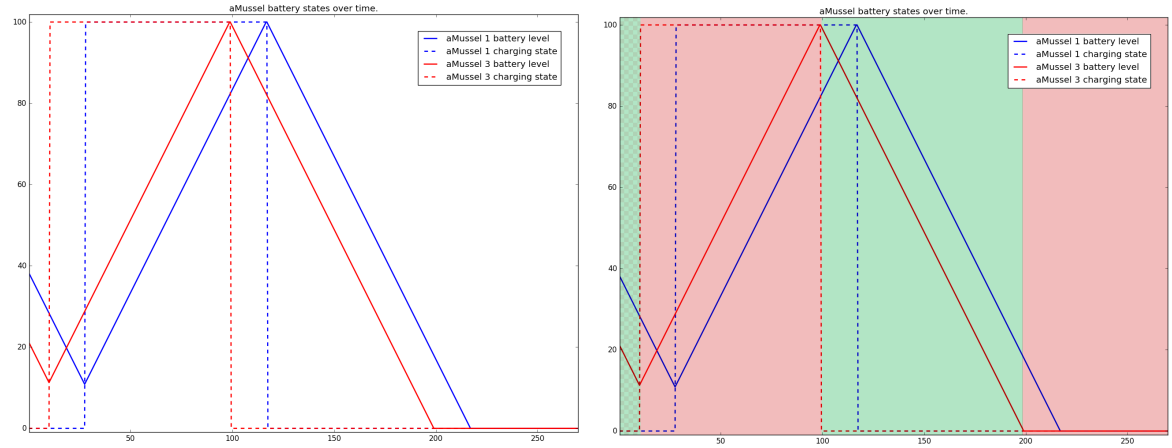


Figure 3.14: 300 time units of simulation of a 3 aPad and 15 aMussel pickup and charging cycle, (dis)charge rate parameters both set to 0.5 - aMussels 1 and 3 shown separately (left), with aMussel 3 uptime overlay (right).

Looking more closely at aMussel 3 in the above example, green signifies an interval where the aMussel is active, while pink means inactive. In the first interval shown in the image, [0 - 10.42], it is possible for the aMussel to be either of the two, as it has requested charging and is waiting for an aPad to come pick it up. It could have sent this request from underwater, acoustically, where it is still active, or via Wi-Fi from the surface, where it is counted as inactive. The aMussel is charging in the [10.41 - 99.16] interval, and is therefore not active in any experiment as it is necessarily on the surface. The aMussel is then undocked at 99.16 and allowed to sink back to the bottom, meaning in the interval [99.16 - 199] it is counted as active - until its battery reaches zero and it presumably shuts down. In a full cycling simulation, at some point before this it would have sent out another charging request and been picked up by an available aPad.

Analysis of the scores and properties of the starting aMussel cluster such as SSE implies the potential of pre-biasing the heuristic selection algorithm in order to start off with some amount of adaptedness to the situation. Depending on the configuration of the swarm (positions, distances, battery states, scatter, etc.), Greedy might be emphasised for its prioritising movement optimisation if the swarm is very scattered, for instance.

An example discrete event simulation output of a single discharge/charge cycle for each aMussel in the system, with a list of events and timestamps as they happen, is given in Listing 3.1.

Listing 3.1: Discrete event simulation log example

```
~STARTING CHARGING SCENARIO~
All docking requests received
aMussel 2 docked at 0.00
aMussel 17 docked at 0.00
aMussel 1 docked at 0.00
aMussel 3 docked at 0.00
aMussel 6 docked at 0.00
aMussel 12 docked at 0.00
aMussel 8 docked at 0.00
aMussel 7 docked at 0.00
aMussel 14 docked at 0.00
aMussel 18 docked at 0.00
Charged aMussel 8 at 63.46
aMussel 8 undocked at 63.46
aMussel 5 docked at 63.46
Charged aMussel 12 at 65.07
aMussel 12 undocked at 65.07
Charged aMussel 6 at 68.30
aMussel 6 undocked at 68.31
Charged aMussel 1 at 68.40
aMussel 1 undocked at 68.40
aMussel 20 docked at 68.40
Charged aMussel 17 at 69.96
aMussel 17 undocked at 69.96
Charged aMussel 7 at 70.48
aMussel 7 undocked at 70.49
aMussel 4 docked at 70.49
Charged aMussel 3 at 72.09
aMussel 3 undocked at 72.10
Charged aMussel 18 at 77.21
aMussel 18 undocked at 77.22
aMussel 19 docked at 77.22
Charged aMussel 2 at 77.42
aMussel 2 undocked at 77.42
Charged aMussel 14 at 77.94
aMussel 14 undocked at 77.94
aMussel 13 docked at 77.94
Charged aMussel 5 at 125.25
aMussel 5 undocked at 125.25
aMussel 11 docked at 125.25
Charged aMussel 4 at 132.98
aMussel 4 undocked at 132.98
aMussel 10 docked at 132.98
Charged aMussel 20 at 133.02
aMussel 20 undocked at 133.02
Charged aMussel 19 at 138.44
aMussel 19 undocked at 138.45
aMussel 15 docked at 138.45
Charged aMussel 13 at 148.68
aMussel 13 undocked at 148.68
aMussel 16 docked at 148.68
Charged aMussel 11 at 196.02
aMussel 11 undocked at 196.02
aMussel 9 docked at 196.02
Charged aMussel 15 at 201.34
aMussel 15 undocked at 201.35
Charged aMussel 10 at 201.80
aMussel 10 undocked at 201.81
Charged aMussel 16 at 222.23
aMussel 16 undocked at 222.23
Charged aMussel 9 at 262.73
aMussel 9 undocked at 262.74
All docking requests handled
~SCENARIO OVER~
```

This pure discrete time event simulation proved a useful stepping stone for conceptualising the system, but it needed to be expanded beyond the capabilities of the tools in order to better represent the heterogeneous robotic system and evaluate its behaviours. For that purpose a ROS-based agent simulation with a vehicle-in-the-loop option was designed and implemented.

3.6 Decision-making proof-of-concept experiment

Experiments were conducted in Biograd na Moru (Figure 3.15) in order to test some of the initial decision-making and agent assignment concepts in development and validate implementation details on actual aPad vehicles. The scenario was designed to include testing mesh network communication, the decision-making and mission execution frameworks,

GPS positioning capabilities, interactions with virtual aMussels, result reporting, and seamless autonomous operation. Trials were first run with five stationary aPad vehicles on land, then with two vehicles in the pool in full movement (Figure 3.16).

Figure 3.15: Google Earth image (Image data: ©2022 CNES/Airbus, Maxar Technologies, image acquired 24/3/2022) showing the Biograd na Moru experiment area - pool and bay.

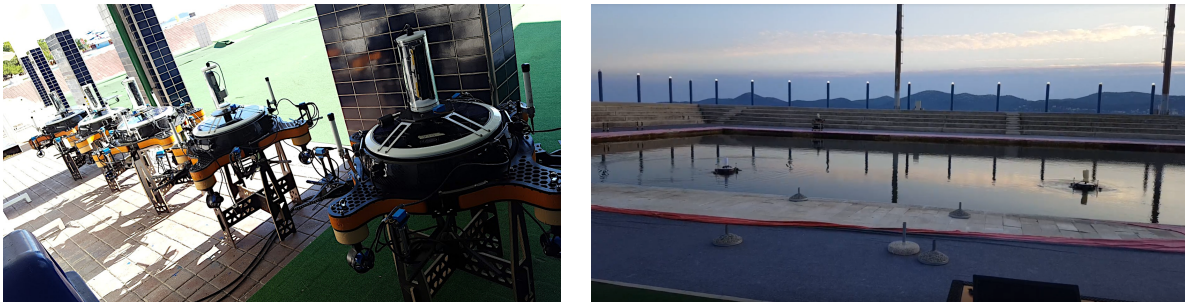


Figure 3.16: Five aPads performing initial communication and decision-making tests on land (left). Two aPads performing decision-making experiment in pool (right).

The aPads conducted autonomous mission planning combining clustering, DE, and two low-level heuristics. To enable an arbitrary number of aMussels in the experiment with little logistical issue, the aMussels were entirely virtual. aPads patrolled and moved around the simulated broadcast positions as if the aMussels were real, while their battery consumption and all processing/calculating/execution times, amount of movement, distance travelled were logged on board the vehicles. One of the main goals of these trials was to see all elements of the aPad system perform in realistic conditions, with such requirements as communication consistency and reliability, and ensuring all aMussels were regularly visited and in the correct order. The full test run procedure is illustrated in Figure 3.17.

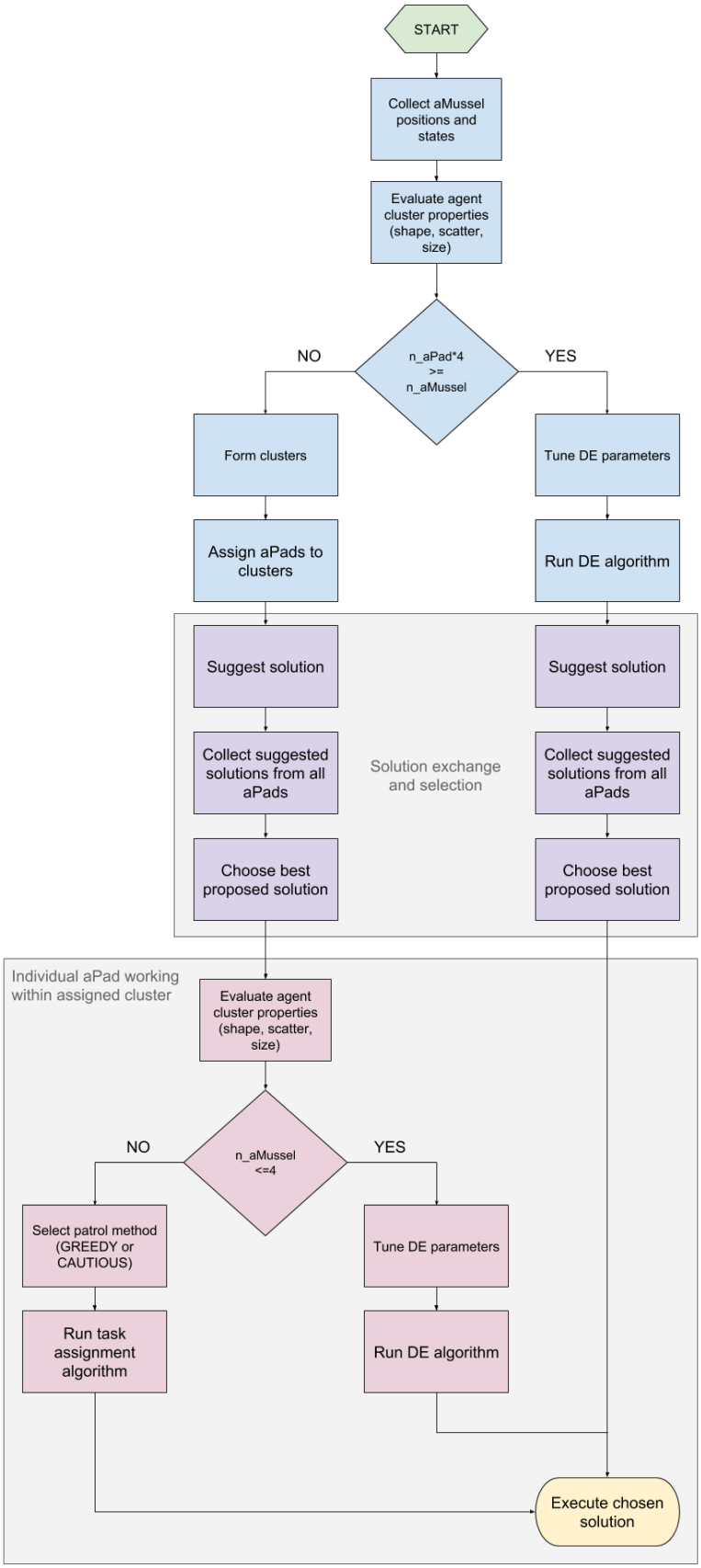


Figure 3.17: Flowchart of the 2018 experiment decision-making scenario.

The aMussel generator node is run first, broadcasting via User Datagram Protocol (UDP) the same JavaScript Object Notation (JSON) packets containing aMussel sensor data as real aMussels, only the position values are randomly generated from within the preset operational area. The aPads collect the generated aMussel charging requests - all data is continuously shared using a UDP bridge unicast between all combinations of pairs of aPads, so there is a smaller chance of one of the aPads being denied certain information packets or some of the aMussels going unheard and uncollected.
Once all the data has been collected, i.e. the aPads have at least an initial matrix of aMussel data and states, the aPads start their mission which includes several steps of decision-making:

- Each aPad considers the initial cluster of present aMussels and forms a solution suggestion which it shares with all other aPads via the payload in its own status message (encoded)
- aPads wait until they have collected solution suggestions from all other involved aPads, or until the predefined timeout happens
- The aPads take (agree upon) the initial solution with the lowest f_{krit} attached (the "best" solution any of the aPads found) and begin work on their own individual segment of it
- Depending on the number of aMussels compared to aPads, the initial solution can be a division into clusters, or a set of routes for each aPad calculated using DE
- If the accepted initial solution was a DE route, each aPad takes the route assigned to it and carries it out
- If the accepted initial solution is a set of clusters, centroids, and aPad-to-cluster assignments, each aPad takes the cluster assigned to it, considers its shape and size, and then proceeds to carry out its patrol within it, choosing from 3 approaches
- aPads execute all required GoTo maneuvers to visit the aMussels in the order they decided upon (for experiments with non-virtual aMussels, they also execute docking and charging procedures)

After the aPads have agreed on an initial solution, i.e. the best solution proposed among them, they work independently within their own assigned clusters in a TSP-like problem set. To aid in the decision-making process, the sizes and shapes of all clusters need to be considered - both the initial "cluster" containing all active aMussels, as well as the clusters calculated during the decision-making steps if the number of aMussels suggests this approach. Depending on the properties of the cluster they have been assigned, the aPads choose between two methods that prioritised different aspects of the swarm - distance and battery state, also known as the Greedy and Cautious heuristics. If the cluster contains fewer than 5 aMussels, DE is applied directly.

DE parameters are tuned as follows. The optimisation equation for the DE algorithm is given in (3.2). A higher K_{charge} gives more weight to the charging/energy transfer component, whereas a higher $K_{movement}$ prioritises travel energy expenditure and gives more weight to the distance-related calculations. For the experiments, a very simple relation was used (3.10):

$$\begin{aligned} K_{movement} &= SSE \\ K_{charge} &= \frac{-1}{SSE} \end{aligned} \tag{3.10}$$

K_{charge} moving towards zero will cause the aPad to disregard battery states and prioritise movement optimisation. $K_{movement}$ going towards zero will cause the aPad to disregard distance optimisation and instead prioritise aMussels in greatest need of charge. The worst case scenario would be all aPad energy used up on movement, none on aMussel charge, whereas the best case scenario is no energy used up on movement, yet all on aMussel charge. All realistically possible cases asymptotically move towards one of the two extremes.

A structure for aPad message payloads containing encoded solution suggestions was constructed in order to enable efficient communication and consensus during collective decision-making. Initially, all five aPads were used in simulated test runs on land - the only difference being stopping the experiment before the actual movement stage. The result of one such experimental run is shown in Figure 3.18, highlighting the messages exchanged between the aPads and showing how fast consensus on a DE solution was achieved - the total time elapsed from the first solution suggestion to the last of the aPads moving to execute was 1 minute and 5.216 seconds.

It was assumed that both the number of active aMussels and aPads in the system was known in advance. The operational area is always known and fixed and corresponds to the outdoor pool used for field trials. While aMussels reported their coordinates in global latitude/longitude, the distances considered are very small (all significantly smaller than 1 km), so calculations were done using Euclidean distances and simplifications with the assumption of a locally flat Earth.

Two aPads were used in pool tests for the complete experiment run, with an example starting state and completed trajectory shown in Figure 3.19. The communication between agents was successful, and all aMussels were approached and "collected" as planned, with the aPads operating fully autonomously. Some interference was present with the GPS localisation on the platforms, leading to oscillation during the start of the aMussel collection step, but this did not noticeably impact algorithm performance.

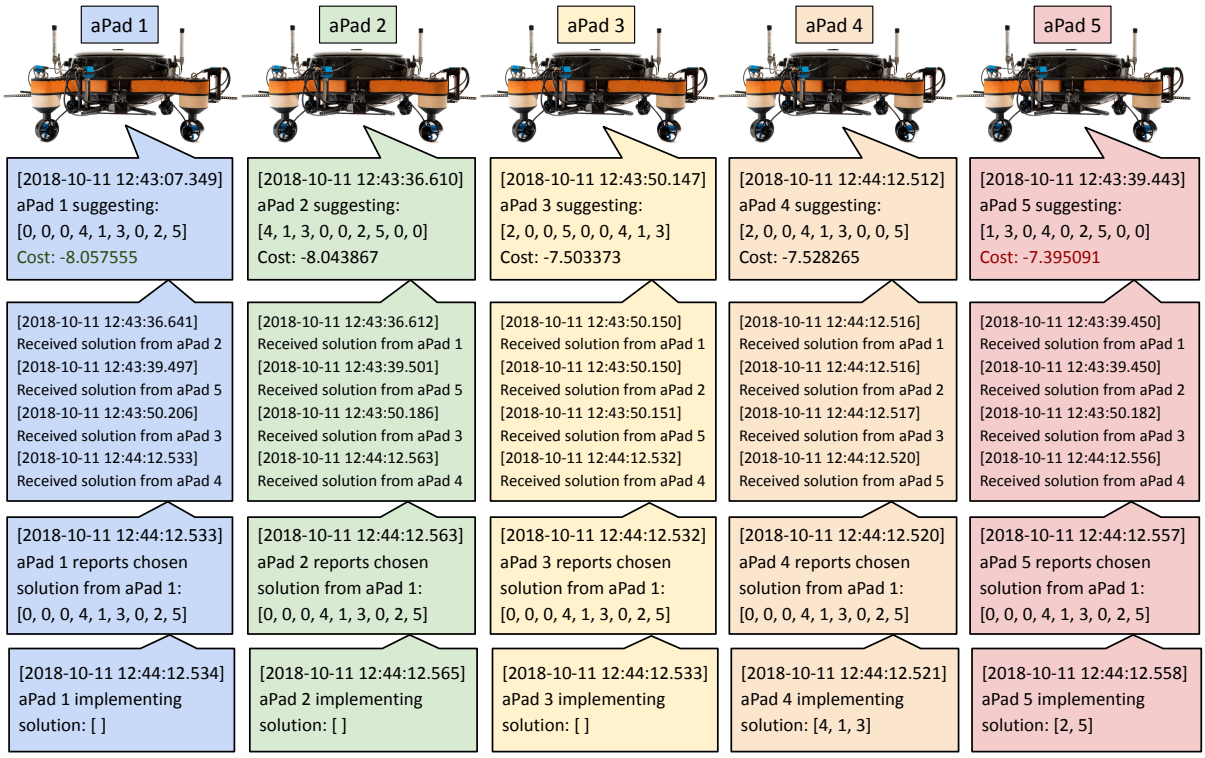


Figure 3.18: Messages exchanged during aPad decision-making in the 2018 proof-of-concept communication-focused experiment.

Two approaches to collecting aMussels within assigned patrol clusters were implemented, corresponding to cluster evaluation based on SSE values: the Greedy and Cautious approaches. In the Greedy approach, the patrolling aPad selects in each step the aMussel closest to the position of the last aMussel it came to collect. The assumption here is that the distances are significant and the aMussels are very scattered. In the Cautious approach, the patrolling aPad selects in each step the aMussel with the lowest battery. The assumption here is that the aMussels are very close together - close enough that optimising based on distances wouldn't give a significant benefit, hence the priority is to charge aMussels that need it most as quickly as possible. Experiments done in a relatively small pool led to aPads most often choosing the Cautious approach to selecting aMussels within their assigned clusters.

In these experiments an initial variant of the decision-making system was used in which one set and final solution for an aPad within its assigned cluster was selected. The next implementation step after this is a step by step approach with the option of fluidly changing between methods, in which a new heuristic/priority is chosen after every step. The step can be after collecting one aMussel, or after every 4 aMussels collected (representing a full "charging cycle" with full aPad capacity), and after each of these steps an analysis is performed in order to evaluate and potentially re-evaluate system performance and decisions.

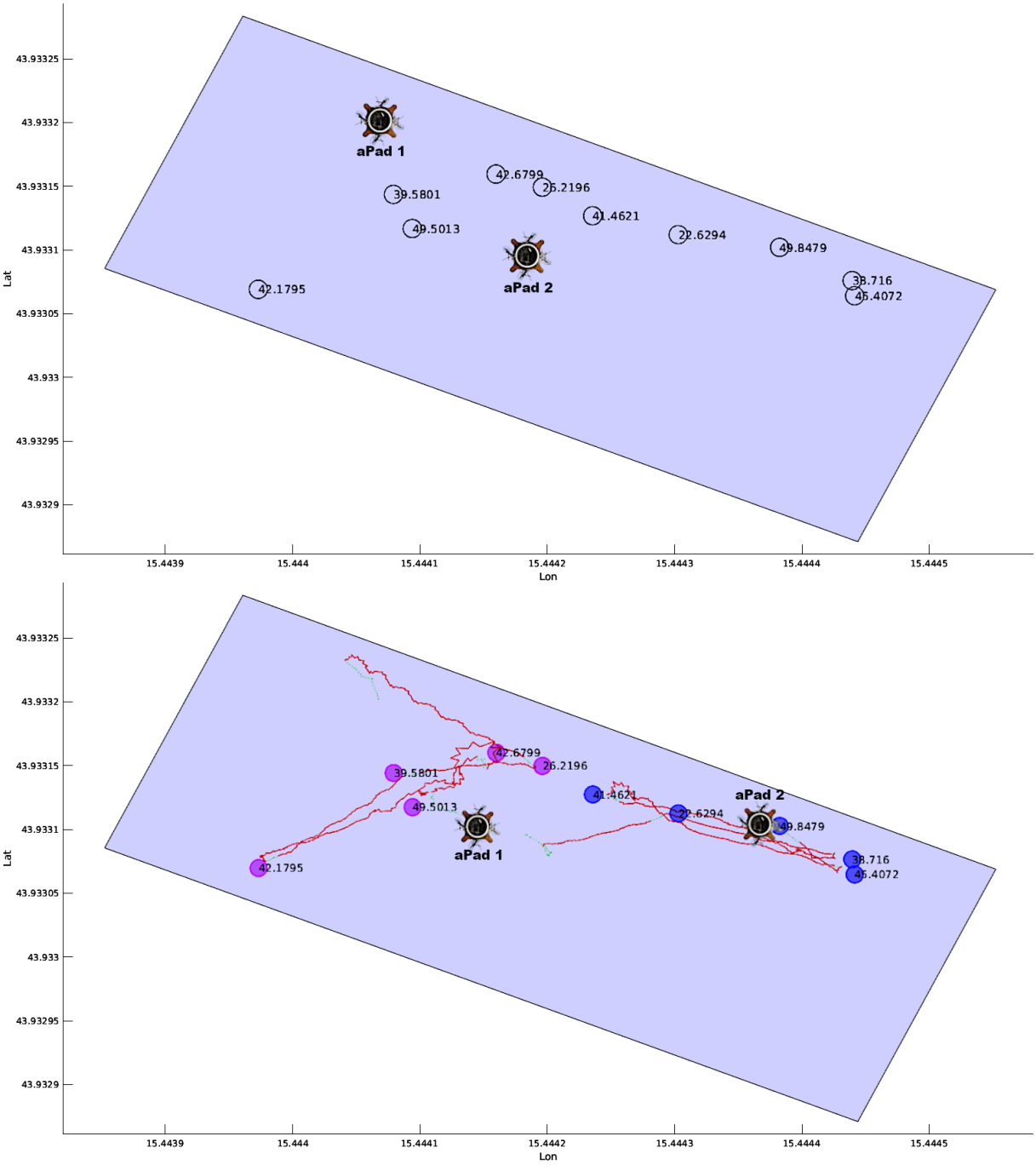


Figure 3.19: An experimental run with two aPads charging 10 aMussels in a small operational area. Initial agent positions with aMussel battery states (top). Final mission execution showing aMussel clustering (blue and magenta) and aPad trajectories (bottom).

Once initial assignment of aMussels to aPads has been achieved and a pool of available heuristics has been established, the next development step includes constructing a validation and evaluation system for solutions and methods which will enable objective grading and aid selection done by the hyper-heuristic decision-making layer.

Chapter 4

High-level heuristics - hyper-heuristics

4.1 Introduction

A general definition of a hyper-heuristic is given in [103] where it is described as an automated methodology for selecting or generating heuristics to solve hard computational search problems. Originally referred to as "heuristics to choose heuristics" [104], it represents a high-level approach that can select and apply to a particular problem instance an appropriate low-level heuristic from a selection pool, and do so at each decision point [105]. Heuristic and metaheuristic methods have been successful in solving real-world computational search problems, however they encounter difficulties in terms of application to newly encountered problems or even new instances of very similar problems. The main cause of these difficulties is the wide range of both algorithm and parameter choices involved in the problem-solving, as well as the lack of guidance as to how to select between them. Additionally, the level of understanding of why various heuristics work in certain situations and not in others may not be adequate to easily and simply make these choices, and difficulty in accurately modelling problems and situations means that strictly mathematically optimal solutions may not actually be the best possible solutions in practice. One of the main goals of using hyper-heuristics is to raise the level of generality at which optimisation systems can operate, leading to the same system being able to address diverse problems such as nurse rostering, university course timetabling, and bin packing [106], [107]. Methods of evaluation have been developed to express how well a hyper-heuristic has generalised (rather than optimised) over a set of problem instances [108].

The framework initially proposed in [104] leads to hyper-heuristics that require very limited information, such as the total number of low-level heuristics present in the system,

whether the problem requires maximisation or minimisation, and the objective or cost function value of the solution being evaluated. In fact, the domain barrier (Figure 4.1) present in the concept of a hyper-heuristic framework prevents the high-level heuristic from ever directly retrieving any information specific to the problem domain, which leads to increased generality of the entire search process. This modular model separates the functionalities of a given problem from the functionalities of the algorithm optimization process [109] [110] [111].

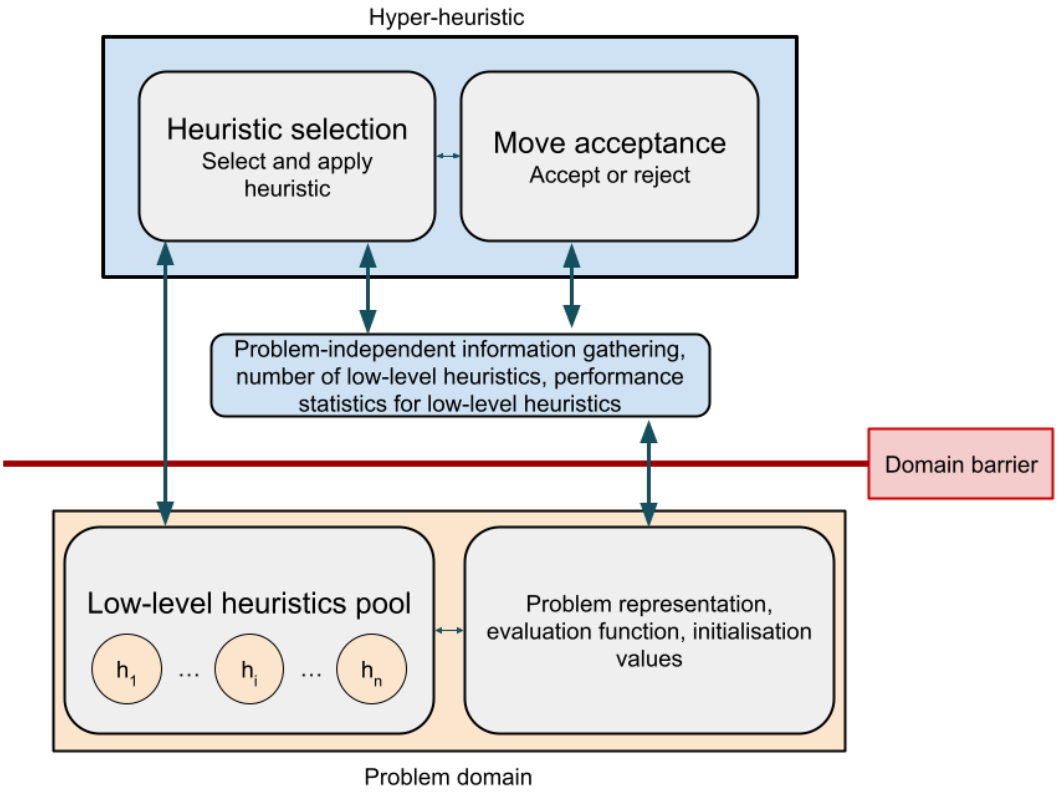


Figure 4.1: General hyper-heuristic framework.

More about hyper-heuristics and specific approaches related to levels of abstraction and problem domain knowledge is given in [112], [113], [114], [115], [116]. An introduction to applying hyper-heuristics to multi-objective optimisation is given in [117].
Hyper-heuristics are commonly divided into three categories based on whether they learn while searching or prior to searching, or if no feedback at all is acquired from the search space: on-line learning, off-line learning, and no learning. They can also be classified as selection or generation hyper-heuristics, depending on whether they select a heuristic among a set of existing heuristics or generate new heuristics from components of the existing low-level heuristics [118] [119]. A visual overview of this classification is shown in Figure 4.2.
The hyper-heuristic employed in this thesis is, according to the above classification, an

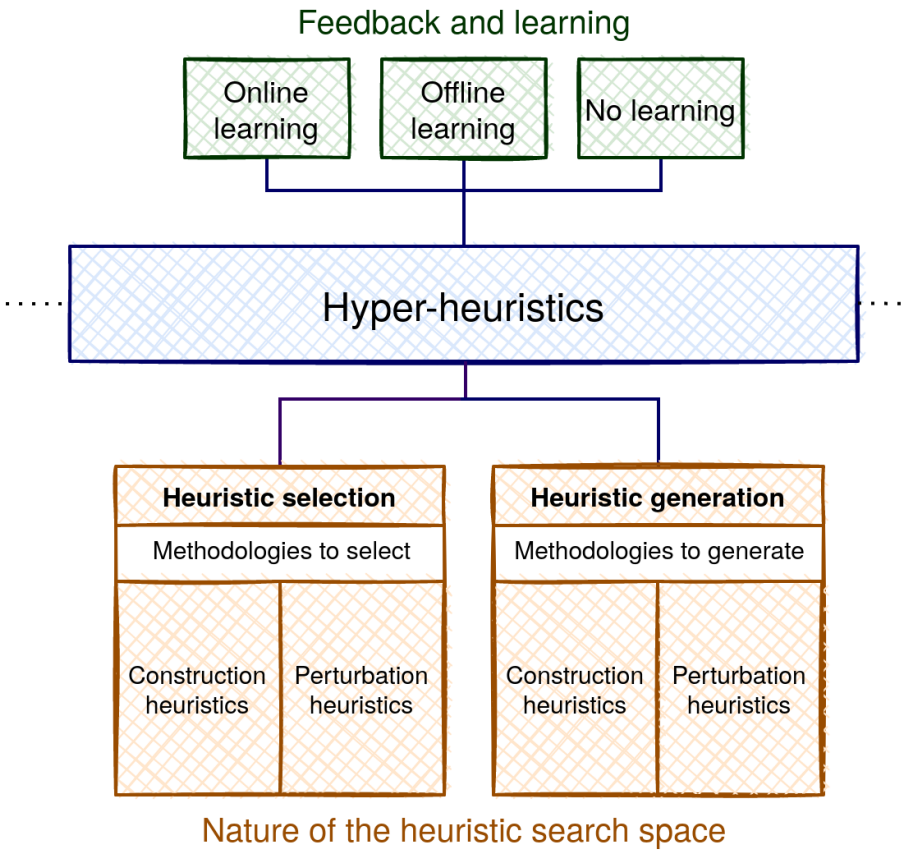


Figure 4.2: Classification of hyper-heuristic approaches.

online learning selection hyper-heuristic. Selection hyper-heuristics deal with problems indirectly, by browsing through a set of available heuristics each search step and choosing which one to apply to the problem at hand based on a history of performance statistics according to a given set of metrics. Two key phases exist in the hyper-heuristic: heuristic selection and move acceptance. The former is the specific method or strategy used for performing the selection from the available pool, while the latter is a binary choice of whether to accept and implement or discard the solution generated by the chosen heuristic, frequently taking the form of All Moves (AM), Only Improving (OI) or Great Deluge (GD) acceptance strategies [120] [121] [122]. In the hyper-heuristic structure in this thesis, changes resulting from the application of a particular heuristic are always accepted (AM) and focus is placed on heuristic selection.

In [123], a hyper-heuristic is presented which starts with random selection, but gradually learns which low-level heuristics perform better and increases their chances to be selected again. This method can produce better solutions than some bespoke metaheuristics, but with far less tuning and oversight and manual adaptation of systems to specific problem instances. A similar approach has been used to create the selection hyper-heuristic implemented in this thesis, based on classical roulette wheel selection.

Low-level heuristics used in hyper-heuristic decision-making structures are often simple operators, but can also be (meta)heuristics themselves. In [124], the authors focus on multi-objective optimisation problems which they seek to address with such metaheuristic selection, and notably include experiments on real world problems and datasets instead of purely abstract function benchmarks in order to evaluate cross-domain performance.

4.2 Heuristic selection, scoring, and evaluation

The roulette wheel selection strategy is frequently used in evolutionary algorithms. Applied to the hyper-heuristic framework, it chooses each heuristic with probability proportional to its weight, or fitness. By modifying these weights based on feedback received from the problem space using several performance indices, online reinforcement learning is introduced into the decision-making system.
After every application of a low-level heuristic, it is scored based on the chosen performance indices. If the heuristic has performed better than the Cumulative Moving Average (CMA) with regards to a certain performance index, it is positively evaluated for that index and its fitness is increased for each success. Otherwise, if a heuristic performs worse with regards to every evaluated performance index, it is considered to have failed and it receives a fixed-step penalty to its fitness.
Hyper-heuristic usage means working separate from the problem domain, with all aspects of the real world abstracted into costs reported back to the hyper-heuristic after implementation and evaluation. If, for example, partway through an experiment a current, wind, or vehicle fault make it difficult for an aPad to move and reach the aMussels chosen in the proposed solution, the movement cost will trend noticeably higher than earlier in the experiment, and the low-level heuristic which produced this solution will be penalised as a result, while one proposing a solution that can work around this disturbance and keep the costs down will be rewarded.
Establishing consistent initialisation for the decision-making system is both important and challenging [125] [126]. During the first iteration, all low-level heuristics have the same fitness value and thus the same chance of being chosen. As an initialisation strategy, for the first four iterations of the decision-making, each of the heuristics will be applied exactly once, in random order, without any scoring or learning taking place. After this step (which also serves to initialise the performance index values and averages), the roulette wheel selection will take over.
Importantly, if the search stagnates and the system enters a local optimum, a selection hyper-heuristic needs to have the ability to select an appropriate low-level heuristic to diversify the search and move to another area of the solution space, even if this causes a

temporary decline in performance. Here it is achieved by enforcing a minimum selection probability threshold for all heuristics, no matter how badly they may have performed in the past. Thus, heuristics that have scored well in the past are more likely to be chosen in the future, while a purely stochastic element remains preserved at all times.
Building on approaches used in [126] and [127], the framework of the implemented reinforcement learning hyper-heuristic can be said to consist of a finite set of states $\mathcal{S}$, an objective or evaluation function $f : \mathcal{S} \to \mathbb{R}$ mapping states to real numbers, a set $H := \{h_1, \ldots, h_n\}$ of n low-level heuristics, and an adaptation scheme $A : \mathcal{S} \times H \to \mathbb{R}$. Reinforcement learning mechanisms assign a positive weight or fitness to each low-level heuristic $i \in [n]$, so $w_i^{(t)}$ denotes the weight of heuristic i in iteration t. As stated earlier, all heuristics are equally likely to be selected at the start, meaning that initially $w_i^{(0)} = w_j^{(0)}$ for all $i, j \in [n]$. The weight of each heuristic is restricted to be within a user-defined interval $[w_{\min}, w_{\max}]$ such that $w_i^{(t)} \in [w_{\min}, w_{\max}]$ for all $i \in [n]$ and $t \geq 0$.
At each iteration t, a heuristic selection strategy is used to decide which heuristic to apply based on a distribution $\mathbf{p}^{(t)} := \left(p_1^{(t)}, \ldots, p_n^{(t)}\right)$. For the roulette wheel selection, the probability can be expressed as (4.1):

$$p_i^{(t)} := \frac{w_i^{(t)}}{\sum_{j=1}^{n} w_j^{(t)}} \tag{4.1}$$

The full reinforcement learning hyper-heuristic is given in Algorithm 1. The adaptation rates α and β are fixed in this implementation, though they can also be variable and functions of current heuristic performance, length of the learning process, or similar.
Due to the nature of the system it is being applied to, the reinforcement learning needs to be able to make conclusions and appropriate fitness adjustments based on relatively sparse feedback, as only once an entire charging solution has been implemented by collecting, charging, and then redeploying the aMussels is useful information about the new state of the system received. There is also no definition of the reward for each possible state of the system proportional to how beneficial that state is to reaching the desired goals [128]. This makes the learning and adjustment process fairly slow, and each full adjustment step more impactful.

4.2.1 Performance evaluation

Wherever possible, calculating the gap between the performance of a method and the known optimal solution is the standard approach to heuristic evaluation [127]. Since an optimal solution is not known here and is computationally too demanding to find, in order to evaluate the performance of the implemented hyper-heuristic, comparisons are done using the best achieved solution.

Algorithm 1 Reinforcement learning hyper-heuristic with roulette wheel selection

Given a finite set $\mathcal{S}$ in the solution space
Let $H := \{h_1, \ldots, h_n\}$ be a set of n low-level heuristics, where $h_i : \mathcal{S} \to \mathcal{S}$.
Let $E := \{f_{e,1}, \ldots, f_{e,k}\}$ be a set of evaluation functions for k performance indices, where $f_{e,j} : \mathcal{S} \to \mathbb{R}$.
Fix $w_{\min}$ and $w_{\max}$ such that $w_{\min}, w_{\max} \geq 0$.
Let α and $\beta \in [0, w_{\max}]$ be a rewarding and punishing rate respectively.
For all $i \in [n]$, set $w_i^{(0)} \geq w_{\min}$.
Select initial heuristic $i \in [n]$, with probability $p_0^{(t)} := \frac{1}{n}$
Let $s^{(0)}$ be the initial solution generated by chosen heuristic i, with $CMA_j^{(0)} = f_{e,j}(s^{(0)})$.
for $t = 1, \ldots, n$ **do**
    Select i from remaining unused heuristics, with probability $p_i^{(t)} := \frac{1}{n-t}$
    $s' := h_i(s)$
    for $j = 1, \ldots, k$ **do**
        $CMA_j^{(t)} = \frac{CMA_j^{(t-1)} + f_{e,j}(s^{t'})}{t}$
    end for
end for
Let τ be the maximum number of steps in the desired mission length.
for $t = n, \ldots, \tau$ **do**
    Select heuristic $i \in [n]$, with probability $p_i^{(t)} := \frac{w_i^{(t)}}{\sum_{l=1}^{n} w_l^{(t)}}$
    $s' := h_i(s)$
    for $j = 1, \ldots, k$ **do**
        $CMA_j^{(t)} = \frac{CMA_j^{(t-1)} + f_{e,j}(s^{t'})}{t}$
        $w_i^{(t+1)'} := \min\left(w_i^{(t)} + \alpha, w_{\max}\right)$ if $f_{e,j}(s^{t'}) \geq CMA_j^{t-1}(f_{e,j}(s^{t-1}))$
    end for
    $w_i^{(t+1)'} := \max\left(w_i^{(t)} - \beta, w_{\min}\right)$ if $f_{e,j}(s^{t'}) < CMA_j^{t-1}(f_{e,j}(s^{t-1}))$for all$f_{e,j} \in E$
    $w_i^{(t+1)} = w_i^{(t+1)'}$
end for

The first benchmark is the most naive baseline - purely random selection, applying a heuristic chosen using a random uniform distribution each step of the experiment. The other comparison benchmarks used are experimental runs in which each of the low-level heuristics present in the system is applied consecutively by itself throughout the duration of the run. The hyper-heuristic selection with reinforcement learning should perform better than simply selecting one of the heuristics and using it continually, or applying the available low-level heuristics at random.

Performance is evaluated based on the four indices presented in Section 3.4: aMussel uptime, aPad movement costs, aMussel coverage/outlier preservation, and uptime distribution (im)balance. In the case of movement cost and uptime imbalance, a lower score is better (implying the movement and energy expenditure of the aPad were minimised, and useful work and uptime were more evenly distributed between all agents), while a higher score is better for outlier preservation and achieved aMussel uptime (meaning measurements were acquired more consistently from distant agents, and aMussels managed to perform more useful work in total).

The various experimental runs are evaluated using ranking methods [129]. The method that finds the best solution in the set of experimental runs being evaluated gets the lowest value, while the worst performing method gets the highest value, with all others placed in between. The method with the overall lowest rank can thus be considered the best performing method [123].

Differential (percent) ranking for heuristic h_i with score v_i with regards to the chosen performance index e_j is calculated as (4.2):

$$d_{i,j}[\%] = 100 \cdot \frac{\|v_{r_{min},j} - v_{i,j}\|}{v_{r_{best},j}} \tag{4.2}$$

Where $v_{i,j}$ is the mean value of the scores heuristic h_i achieved in performance index e_j throughout the length of the experiment, and $v_{r_{best}}$ is the mean score value the best-scoring heuristic achieved (minimum rank r_{min} is the equivalent of best rank r_{best}). Thus, this score represents the distance of each heuristic score from the best score achieved in the experiment. The best achievable rank is 1, while the best achievable differential rank is 0.

Total score ranking for each heuristic is done by scaling the $v_{i,j}$ means to a $[0, 100]$ interval for each heuristic h_i and performance index e_j of N_{index} indices, then calculating a sum of scores achieved for each index, (4.3):

$$S_{total,i} = \sum_{j=1}^{N_{index}} v_{i,j,scaled} \tag{4.3}$$

For performance indices where a lower score is better, an inverse of the $v_{i,j}$ mean is passed

to the scaling step. As there are four performance indices taken into account, the highest achievable total score is 400.

4.3 Hyper-heuristic decision-making simulations

Four simulated experiments/benchmark scenarios are presented in the following subsections: a baseline simulation of a swarm with no disturbances or outliers, two instances in which a disturbance occurs partway through the experiment leading to a change in agent capabilities and thus behaviours, and one in which an aMussel is deployed a considerable distance away from the rest of the swarm and is thus a location-based outlier. Simulated mission length for all experiments is 129600 seconds, which, with the employed time scale of 30, is the equivalent of 45 days of aMussel operation. The window size for CMA calculation is set to 50. The Random low-level heuristic was not present in the heuristics pool for these simulations, due to the fact the roulette wheel selection itself provides the desired stochastic component.
All runs of the same experiment case have the same starting conditions (aPad position, aMussel positions, aMussel battery states). Thus, all runs that use a single low-level heuristic are essentially deterministic. The roulette wheel and random selection runs were performed multiple times and the results presented here were selected on a "best of five" basis. In these examples all scores are equally weighted: all performance indices are considered of equal importance. This can be modified - for example, if preserving outliers is considered a more important feature to reward, or if there is a need for the system to primarily focus on the aPad's energy expenditure.

4.3.1 Venice baseline simulation

The aMussels are distributed as shown in 4.3 in a simulated Venice lagoon locale. Both aMussel positions and battery states were initially determined using a random uniform distribution. No disturbance occurs during the entire length of the mission and there are no adverse environmental effects present, allowing the swarm to operate in an ideal state. Figure 4.4 shows the CMA values of the four performance indices for each of the experimental runs throughout the length of the experiment. This represents the historical data to which the hyper-heuristic compares the latest performance of each selected heuristic during the evaluation and scoring step of the process. The strengths and weaknesses of each method present in the system are already clearly indicated in the trends. The fourth performance index, uptime distribution imbalance, is calculated based on data on the total achieved aMussel uptime per agent and by heuristic, as shown in Figure 4.5.
For the purposes of ranking, means of the achieved scores for all methods are calculated.

Figure 4.3: Google Earth image (Image data: ©2022 CNES/Airbus, Maxar Technologies, image acquired 30/07/2022) showing the aMussels in the simulated Venice experiment area.

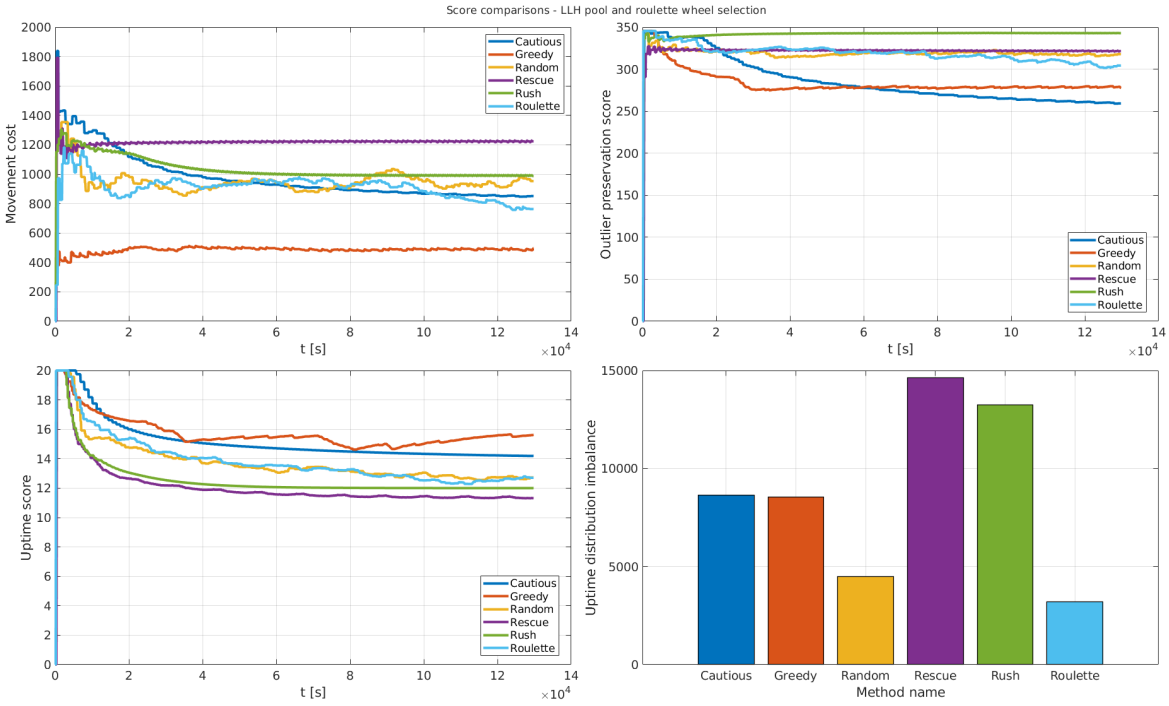


Figure 4.4: Comparison of scores during experimental runs in baseline scenario using only one heuristic, versus roulette wheel selection and random selection.

Figure 4.6 shows a comparison of the results with the best- and worst- performing instances for each performance index highlighted in green and red, respectively. Figure 4.7 shows the final ranking results, as well as a comparison of differences between scores achieved by each heuristic and the best achieved score, with an overview provided in Table

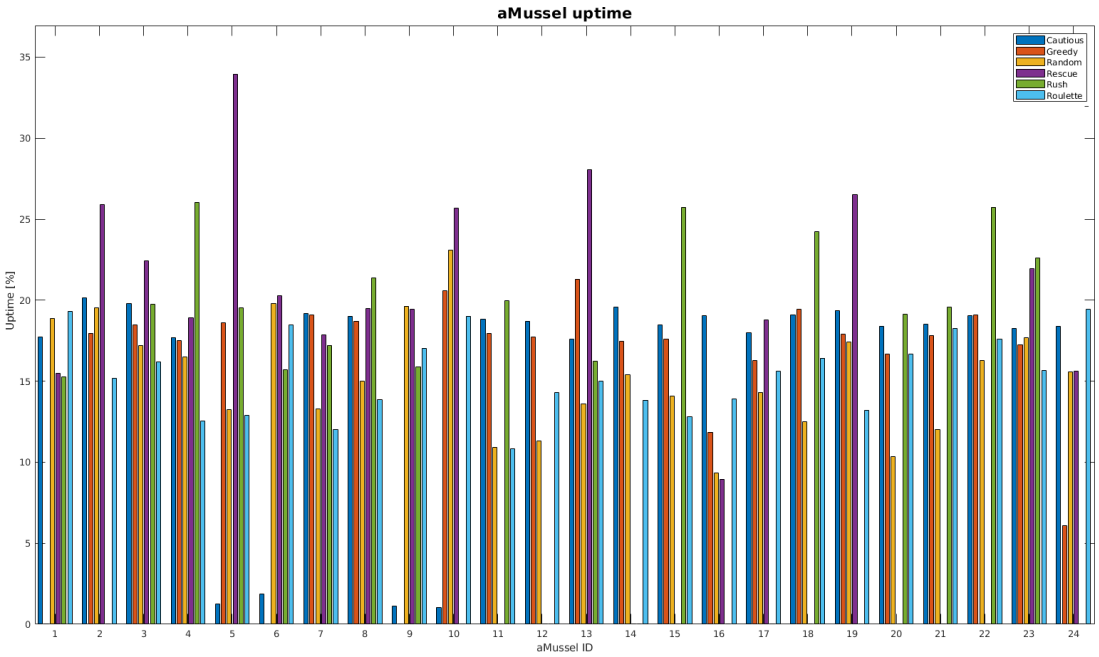


Figure 4.5: Distribution of aMussel uptime in baseline scenario by agent ID in each of the experimental runs.

4.1.

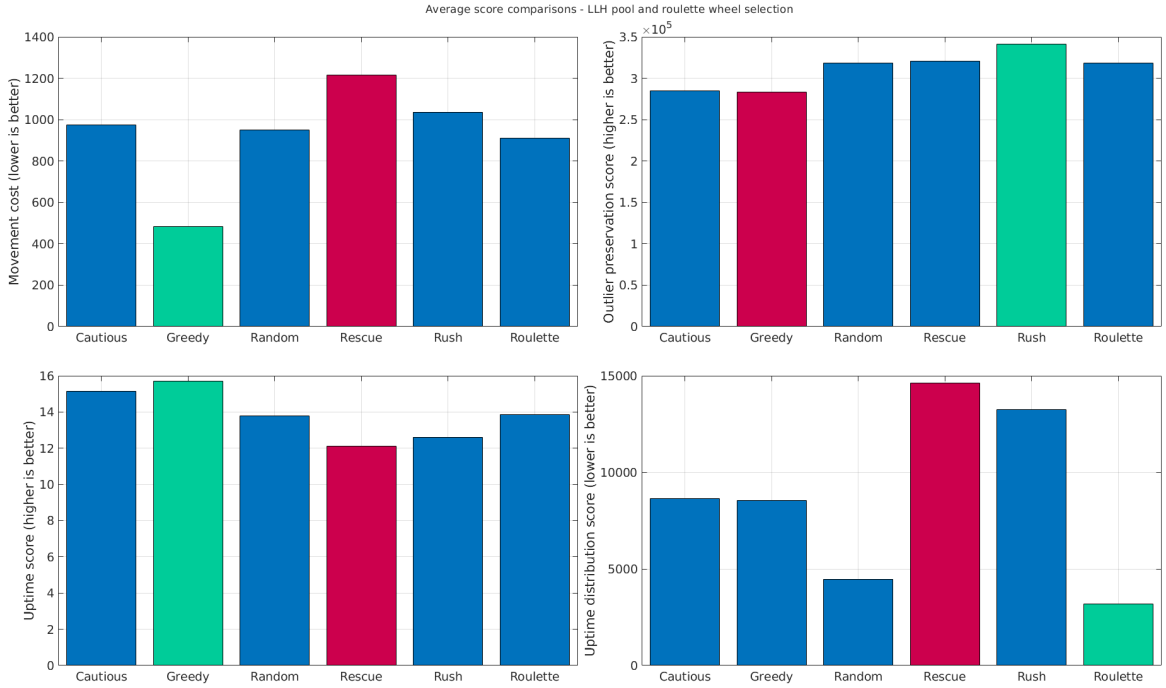


Figure 4.6: Comparison of means of scores achieved by each heuristic in baseline scenario using only one heuristic, versus roulette wheel selection and random selection. Worst-performing heuristic shown in red, best-performing shown in green.

A spider chart breakdown of scores achieved by each heuristic as well as the two heuristic

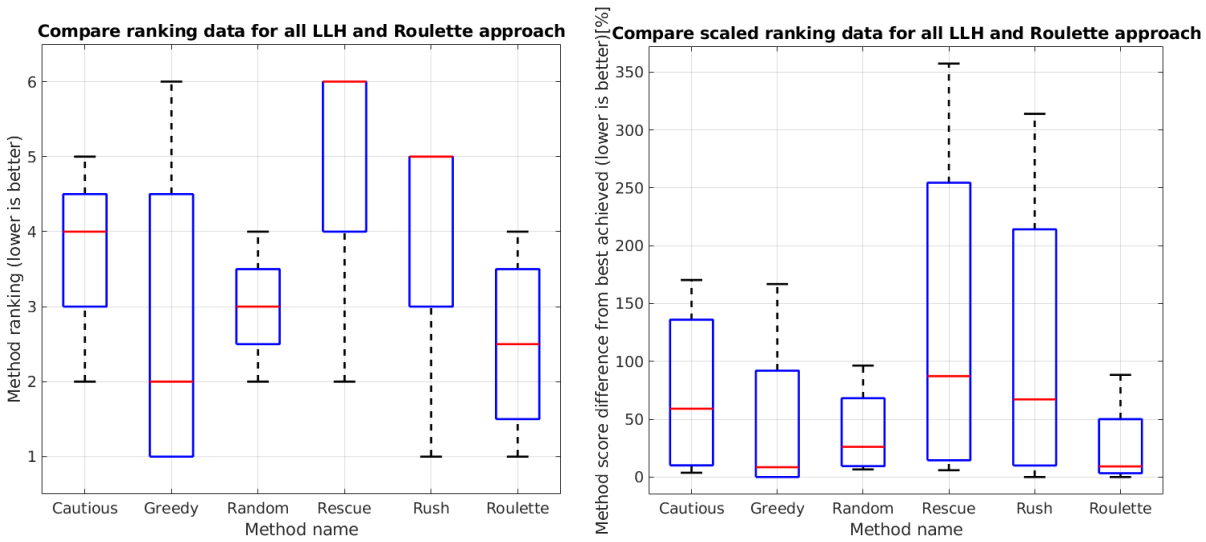


Figure 4.7: Comparison of ranks (left) and differential percentage ranks (right) achieved by each heuristic and roulette wheel selection in baseline scenario.

Table 4.1: Scores and ranking - Venice baseline simulation.

Method	Median [%]	Min [%]	Max [%]	Average Rank	Total score
Cautious	59.062	3.7023	170.297	4	125.35
Greedy	8.4816	0	166.8188	2	221.78
Random	26.1115	6.5864	96.3252	3	191.82
Rescue	87.1974	5.9036	357.3709	6	68.55
Rush	67.1207	0	313.9039	5	130.58
Roulette	9.1858	0	88.263	2.5	233.79

selection methods and ranking of total scores in descending order are given in Figure 4.8, highlighting the strengths and weaknesses of each approach. The Greedy heuristic is, as expected, superior when it comes to optimising for movement efficiency, while the Rescue method's primary strength is outlier preservation. The two runs in which multiple heuristics are applied (roulette wheel and random selection) have both shown they provide a more even distribution of uptime over all aMussels in the system, while Greedy, Rush, and Rescue heuristics instead focus on a limited group of agents.

A closer look at the roulette wheel selection hyper-heuristic run follows, with the total count of times each low-level heuristic was chosen and applied shown in Figure 4.9. It also shows how these choices were distributed over the scenario's duration, as well as how the fitness values for each method changed over time. The fitness scores are displayed as probabilities which are equal to 1 in sum and represent the relative sizes of the slices of the roulette wheel in each time step of the experiment. This means it is possible for

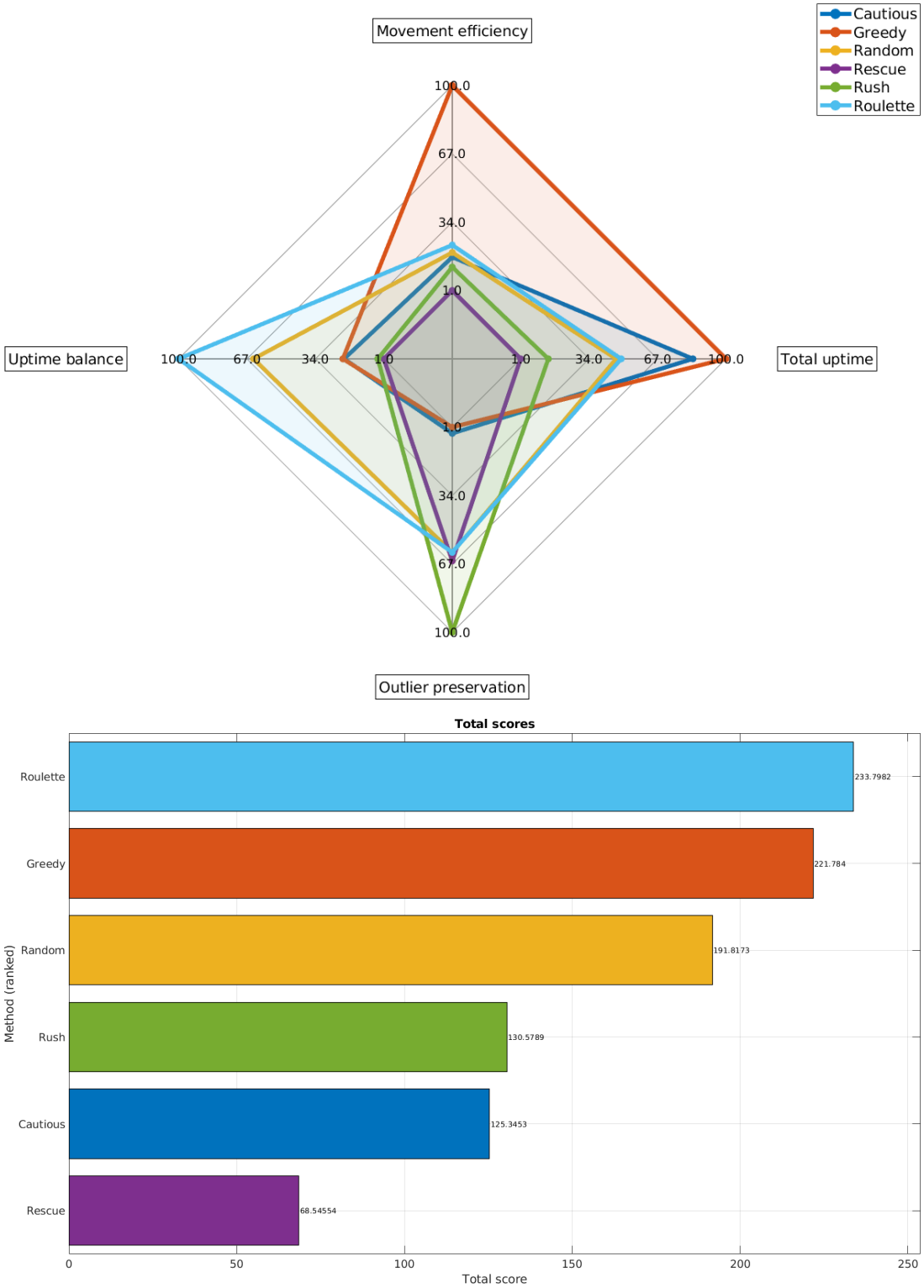


Figure 4.8: Comparison of scores achieved (top) and total score ranking (bottom) for baseline scenario.

a method's fitness to go down even when it is not being evaluated, as another method being rewarded increases its share of the roulette wheel and reduces all others.

In the experiment presented, the Greedy method was used the most and offered good performance judging by its consistently high evaluation and rising fitness, while Rescue

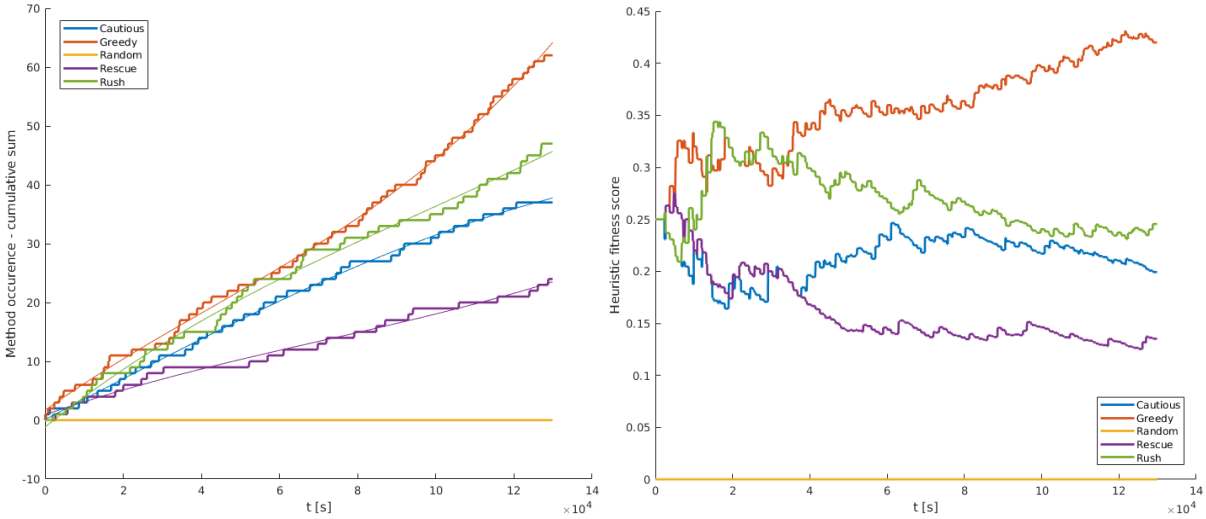


Figure 4.9: Occurrence numbers of each heuristic in the roulette wheel selection (left) and heuristic fitness (right) over time for baseline scenario.

was used the least. Its presence did, however, visibly serve to ensure better outlier coverage than if only the otherwise excellent-performing Greedy method had been used alone, highlighting one of the original motivations behind the hyper-heuristic roulette wheel approach.

4.3.2 Thruster failure/disturbance one third through mission

In this scenario, the aMussels are distributed as shown in 4.3 in a simulated Venice lagoon locale. Once one third of the total mission time has elapsed, a disturbance happens in the form of aPad "thruster failure", resulting in movement speed dropping considerably (10 times) and all movement times and costs proportionately rising. The effects of this disturbance on swarm performance are studied in order to demonstrate the benefits of online reinforcement learning and show the adaptability the learning selection hyper-heuristic gives the system.

Figure 4.10 shows the values of the performance indices for all heuristics throughout the length of the experiment, clearly displaying the universal effects of the disturbance on system performance by any criteria. The total achieved aMussel uptime per agent is shown in Figure 4.11. The Greedy method is most noticeably impacted, which is to be expected since it doesn't work with any system state information other than physical distances and thus has no mechanism of addressing its movement-caused deteriorating performance in any way.

Figure 4.12 shows the final means of achieved scores for all methods, with the best- and worst- performing instances highlighted. Figure 4.13 shows a comparison of differences between scores achieved by each heuristic and the best achieved score, with an overview

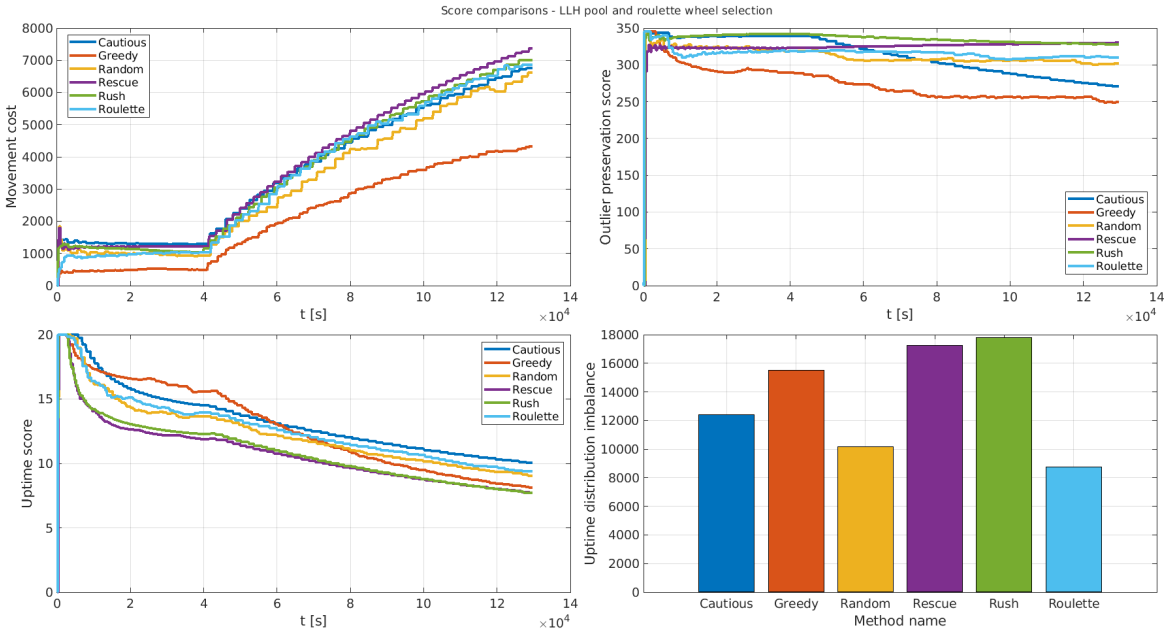


Figure 4.10: Comparison of scores during experimental runs in disturbance scenario using only one heuristic, versus roulette wheel selection and random selection.

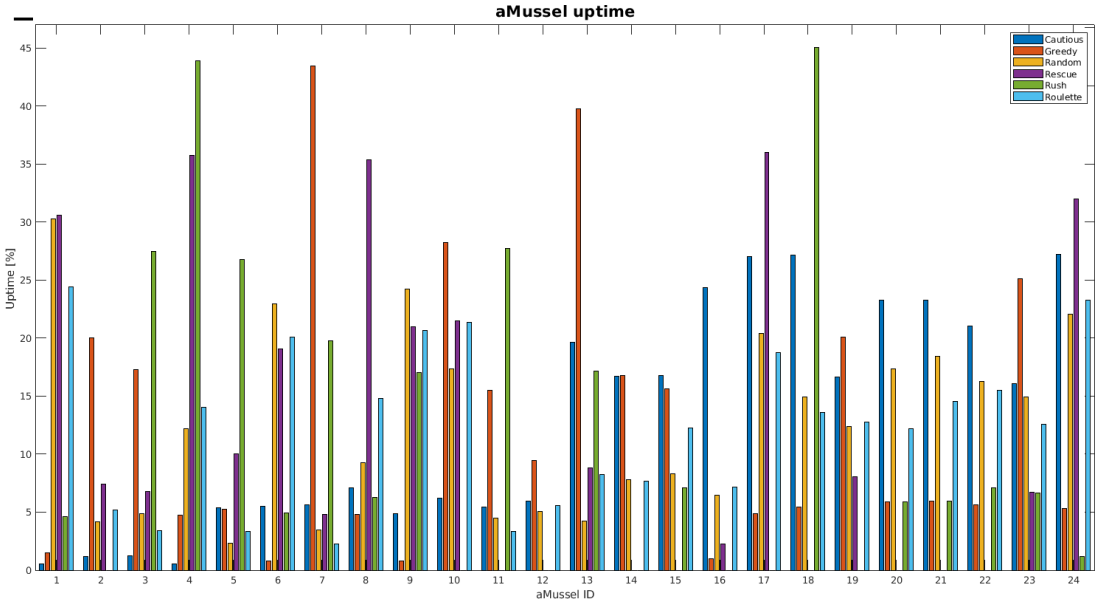


Figure 4.11: Distribution of aMussel uptime in disturbance scenario by agent ID in each of the experimental runs.

provided in Table 4.2.

A spider chart breakdown of total scores achieved by each heuristic as well as the two selection methods is shown in Figure 4.14, followed by a ranking of total scores in descending order in Figure 4.15. Once again, the specific strengths and weaknesses of each low-level heuristic can be clearly seen, as well as the benefits of combining different heuristics in a single run. The Cautious method's focus on keeping all aMussels charged, no matter how

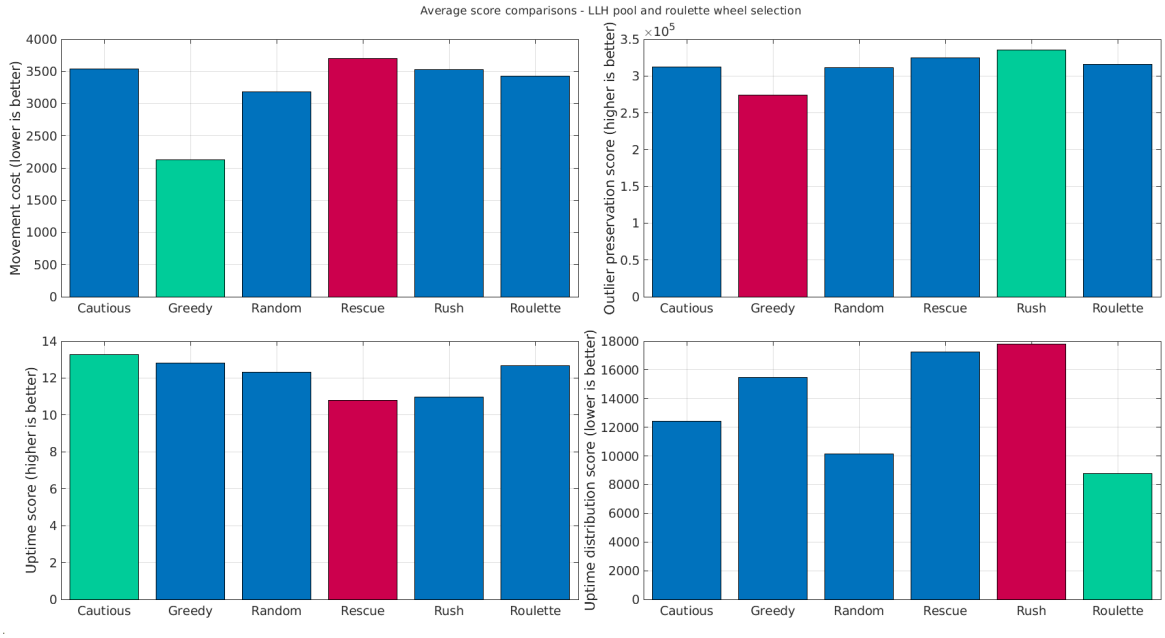


Figure 4.12: Comparison of means of scores achieved by each heuristic in disturbance scenario using only one heuristic, versus roulette wheel selection and random selection. Worst-performing heuristic shown in red, best-performing shown in green.

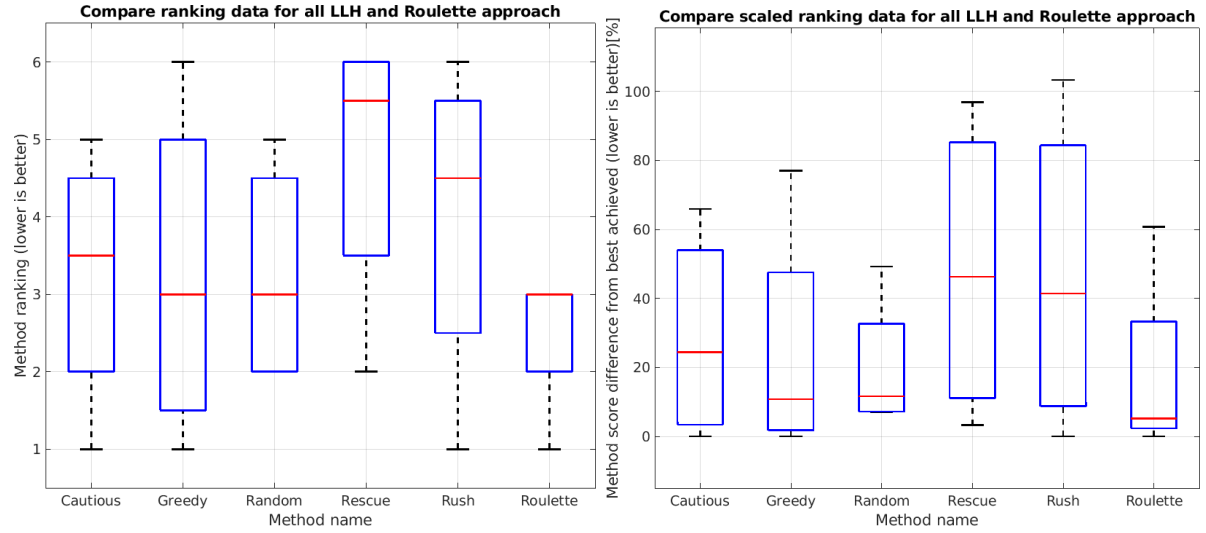


Figure 4.13: Comparison of ranks (left) and differential percentage ranks (right) achieved by each heuristic and roulette wheel selection in disturbance scenario.

distant, is visible in how well it performs with regards to total achieved uptime, as well as achieving the best uptime balance score of all the low-level heuristics.

Looking more closely at the roulette wheel experiment, the total count of times each low-level heuristic was chosen and applied by the roulette wheel selection hyper-heuristic is shown in Figure 4.16. Also shown is the distribution of these choices in the scenario's duration, as well as how the fitness values for each method changed. The moment of disturbance is clearly indicated.

Table 4.2: Scores and ranking - Venice disturbance simulation.

Method	Median [%]	Min [%]	Max [%]	Average Rank	Total score
Cautious	24.3356	0	65.9361	3.5	212.75
Greedy	10.7833	0	76.9925	3	198.05
Random	11.6429	7.0316	49.189	3	219.71
Rescue	46.2947	3.202	96.8368	5.5	88.63
Rush	41.3683	0	103.3228	4.5	117.67
Roulette	5.187	0	60.705	3	256.17

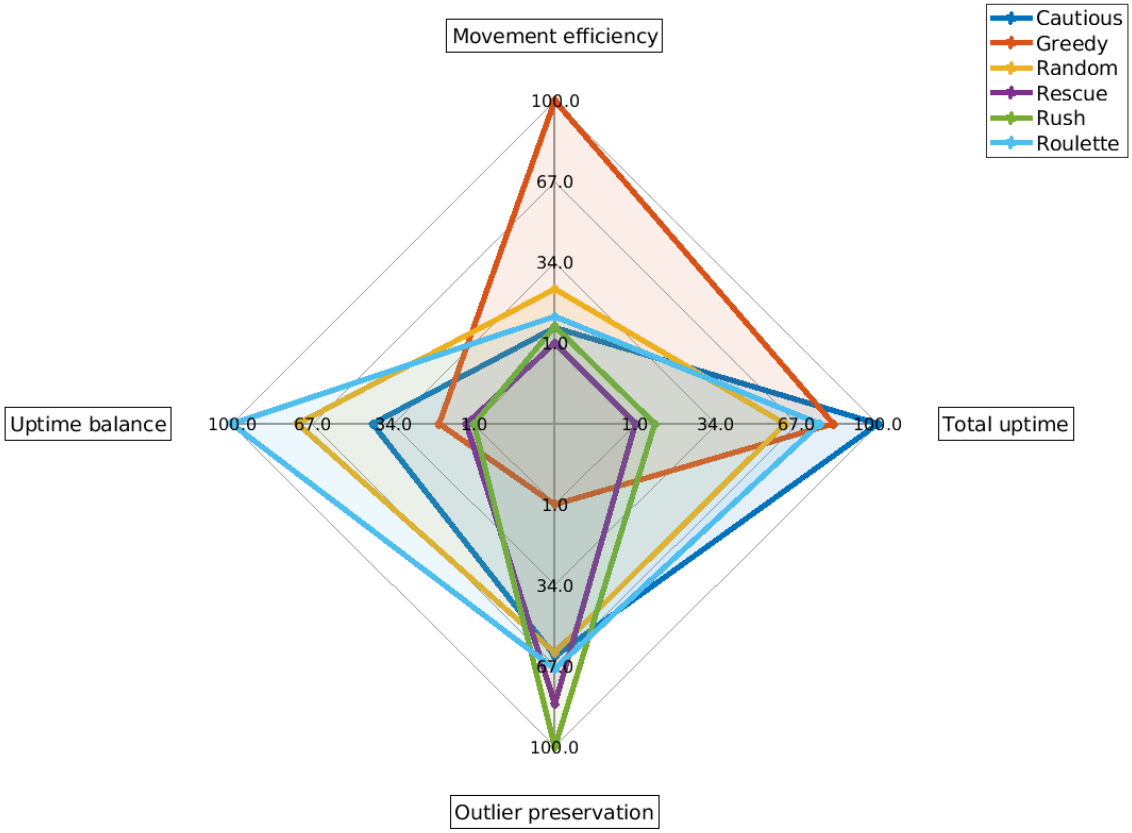


Figure 4.14: Comparison of total scores achieved in the disturbance scenario.

In this specific scenario, Rescue is the only method that achieves an upward trend in fitness post-disturbance - with the aPad's movement severely impeded, it is impossible to achieve an improvement in total uptime or movement cost reduction, and so the aPad resorts to maintaining the edges of the swarm, since it is still possible to successfully preserve outliers. Greedy is extremely efficient with regards to movement and provides a good boost to uptime in the aMussels it picks up - however, it represents a type of local

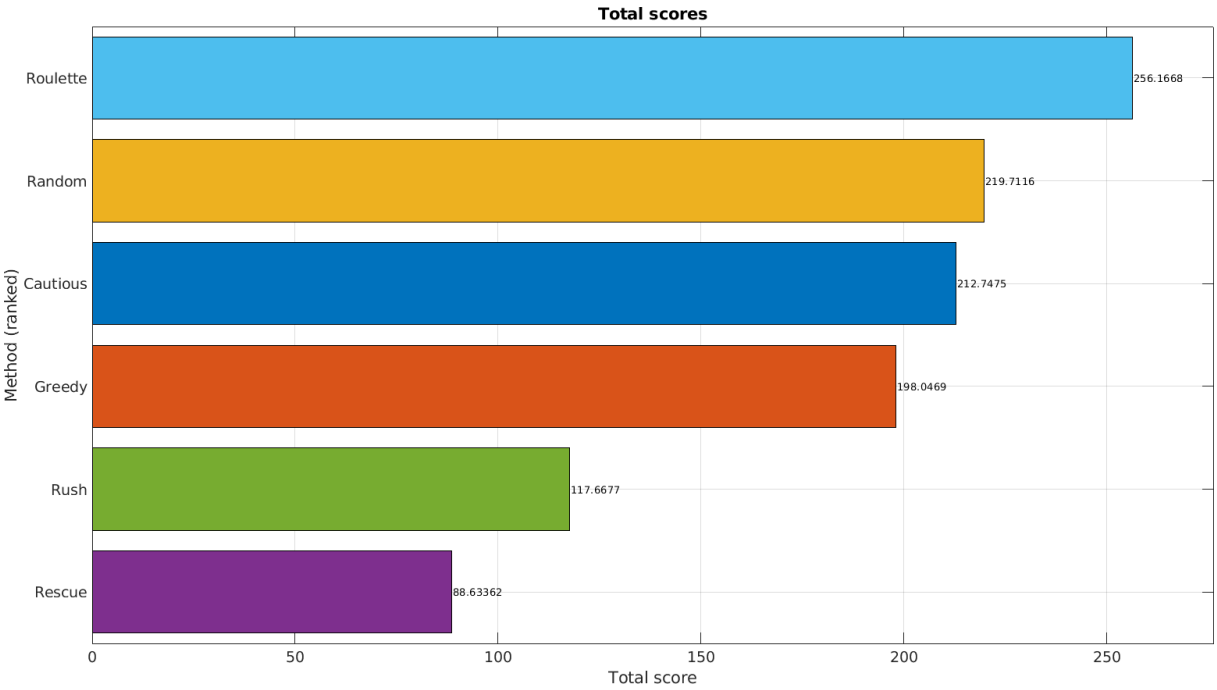


Figure 4.15: Total score ranking for disturbance scenario.

minimum and rarely moves beyond one group of agents, leading to the previously noted pronounced impact post-disturbance.

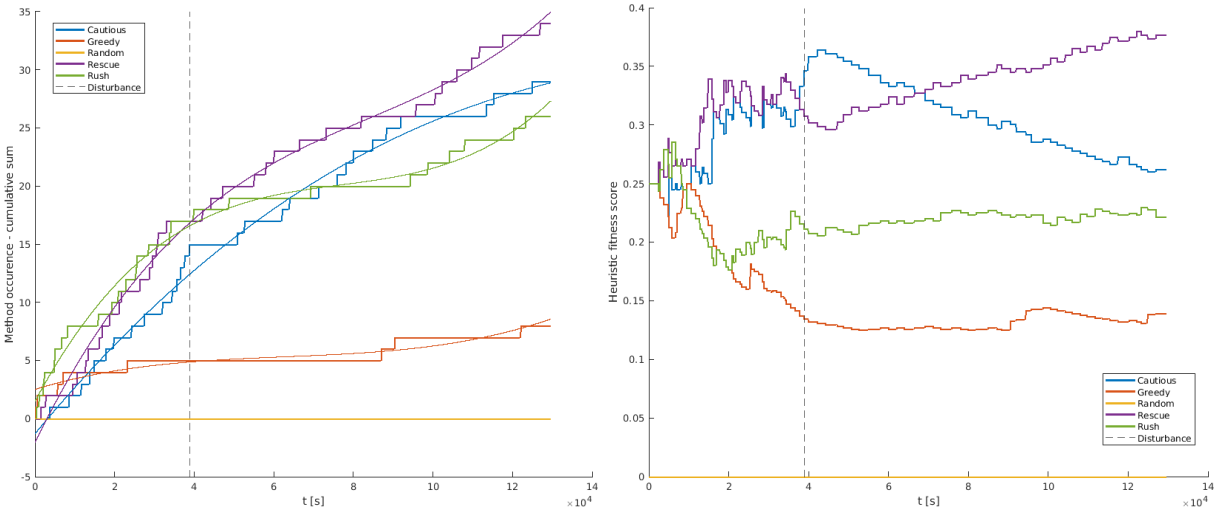


Figure 4.16: Occurrence numbers of each heuristic in the roulette wheel selection (left) and heuristic fitness (right) over time for disturbance scenario.

4.3.3 Thruster failure/disturbance halfway through mission

This experiment uses the same map as the previous simulation, however the disturbance happens later in the run. This, combined with the reduced speed of the aPad, means the system has a much shorter time to adapt to the new conditions, and even fewer

chances to evaluate the performance of each heuristic. Figure 4.17 shows the values of the performance indices for all heuristics throughout the length of the experiment, once again clearly displaying the universal effects of the disturbance on system performance by any criteria. The total achieved aMussel uptime per agent is shown in Figure 4.18. The Greedy method remains the most noticeably impacted.

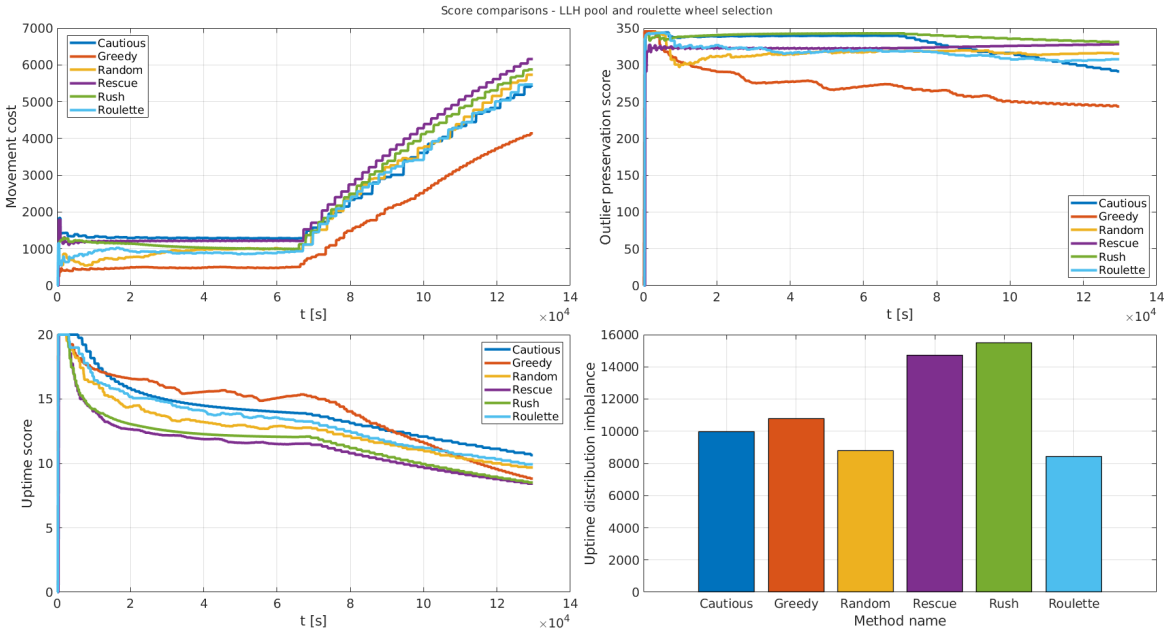


Figure 4.17: Comparison of scores during experimental runs in late disturbance scenario using only one heuristic, versus roulette wheel selection and random selection.

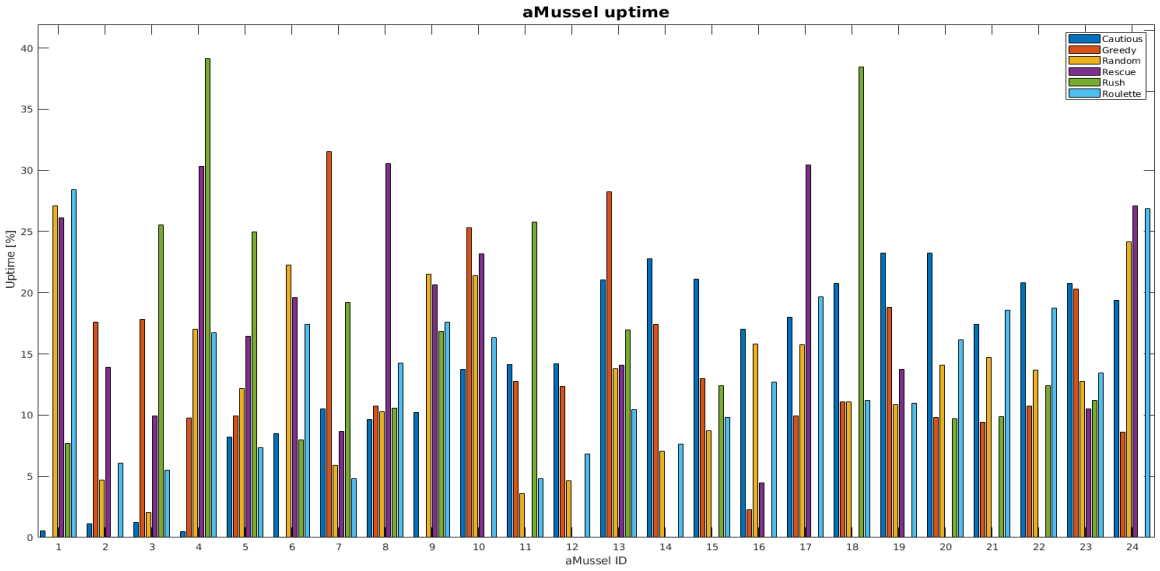


Figure 4.18: Distribution of aMussel uptime in late disturbance scenario by agent ID in each of the experimental runs.

Figure 4.19 shows the final means of achieved scores for all methods, with the best- and

worst- performing instances highlighted. Figure 4.20 shows a comparison of differences between scores achieved by each heuristic and the best achieved score, with an overview provided in Table 4.3.

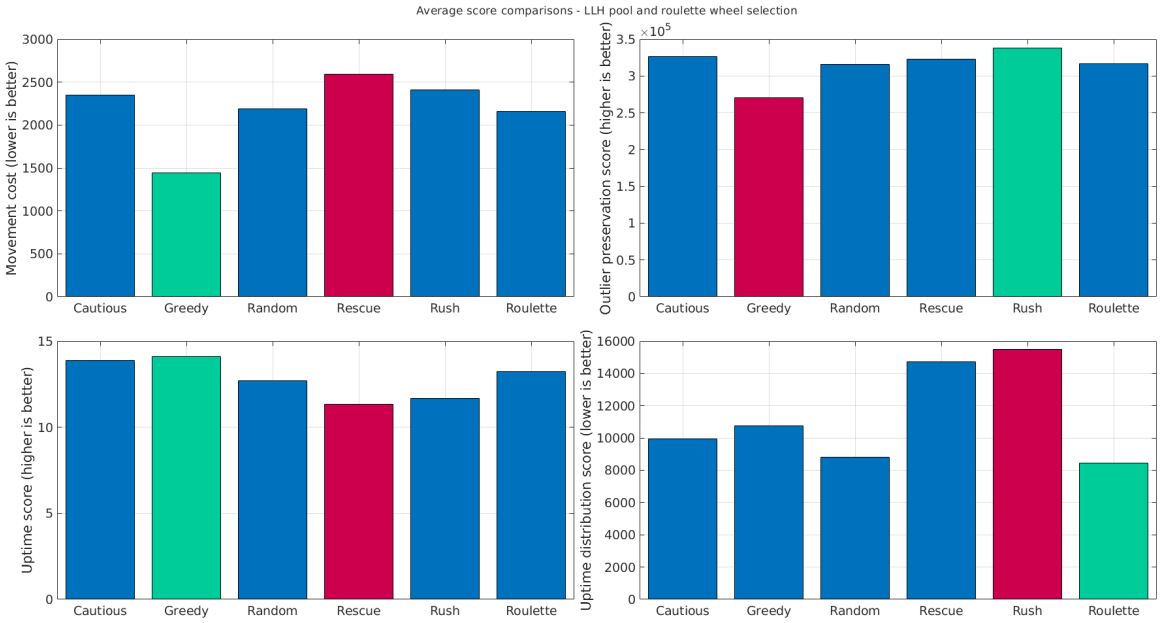


Figure 4.19: Comparison of means of scores achieved by each heuristic in late disturbance scenario using only one heuristic, versus roulette wheel selection and random selection. Worst-performing heuristic shown in red, best-performing shown in green.

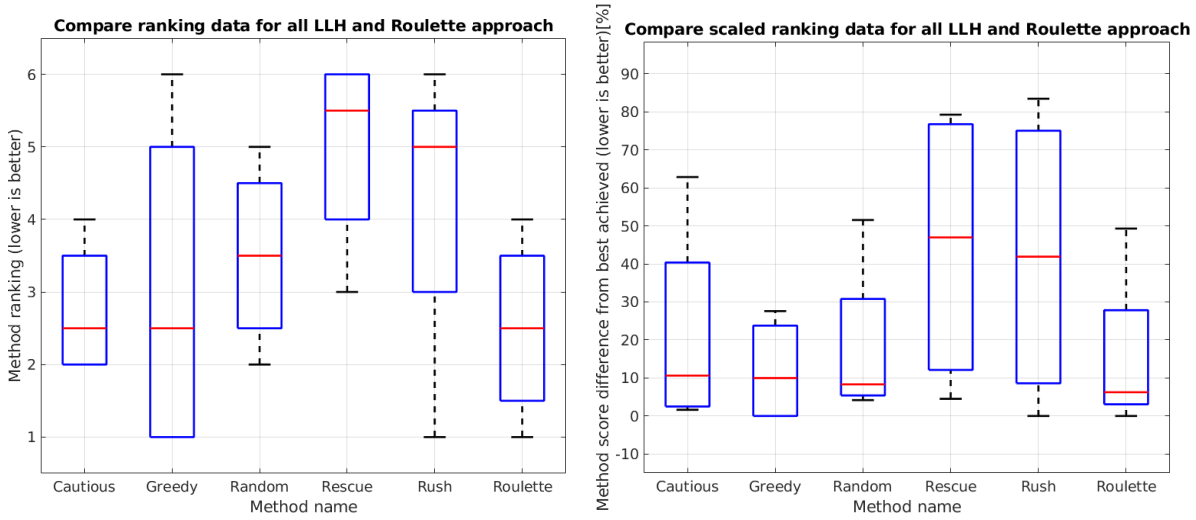


Figure 4.20: Comparison of ranks (left) and differential percentage ranks (right) achieved by each heuristic and roulette wheel selection in late disturbance scenario.

A spider chart breakdown of total scores achieved by each heuristic and the two selection methods, along with a corresponding ranking of all the methods by total scores in descending order is given in Figure 4.21 and Figure 4.22.

Table 4.3: Scores and ranking - Venice late disturbance simulation.

Method	Median [%]	Min [%]	Max [%]	Average Rank	Total score
Cautious	10.6181	1.6314	62.8506	2.5	255.71
Greedy	9.9719	0	27.576	2.5	253.95
Random	8.3243	4.1726	51.5476	3.5	231.89
Rescue	46.9772	4.5187	79.2593	5.5	86.84
Rush	41.9042	0	83.4551	5	125.13
Roulette	6.2476	0	49.3032	2.5	263.67

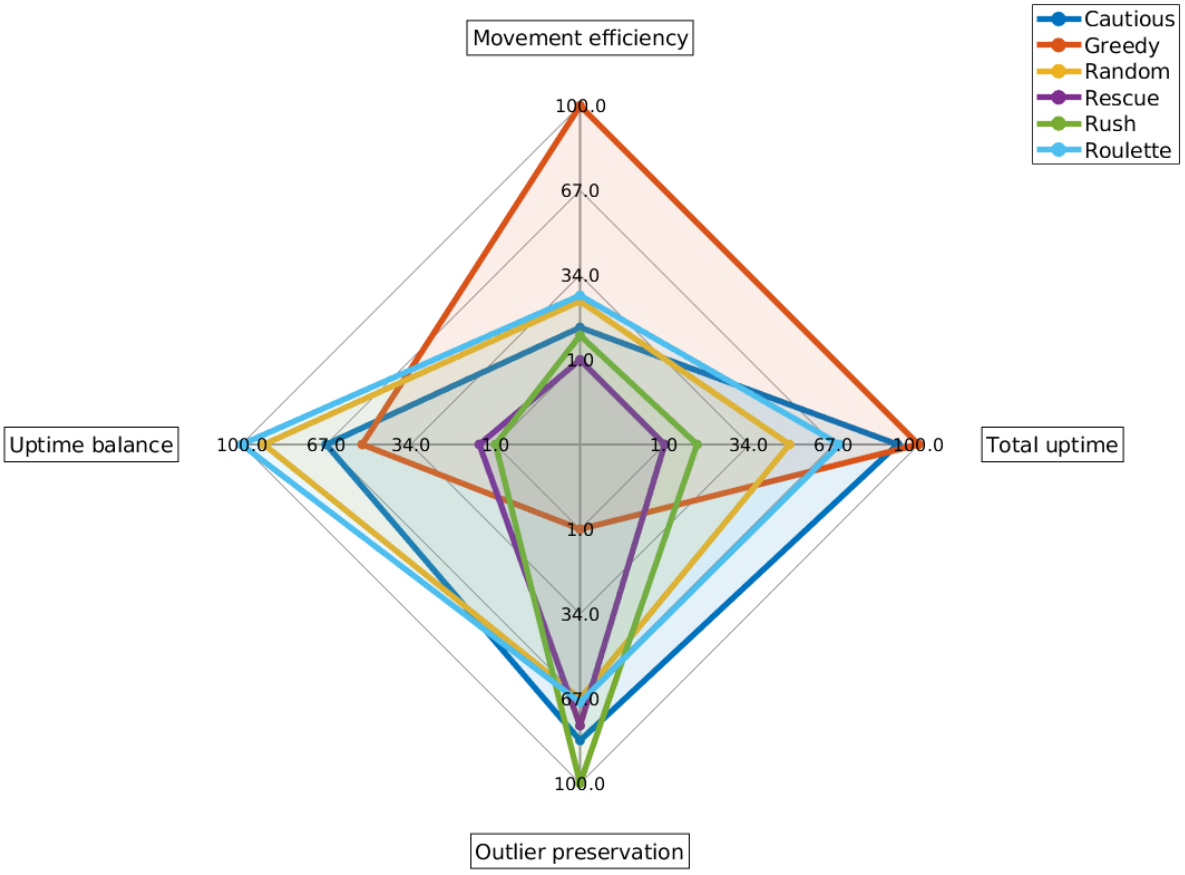


Figure 4.21: Comparison of total scores achieved in the late disturbance scenario.

A total count of times each low-level heuristic was chosen and applied by the roulette wheel selection hyper-heuristic is shown in Figure 4.23, along with how these choices were distributed in the scenario's duration, as well as how the fitness values for each method changed. In this scenario, the performance of methods chosen before the disturbance carries more weight, however similar trends post-disturbance as in the earlier disturbance

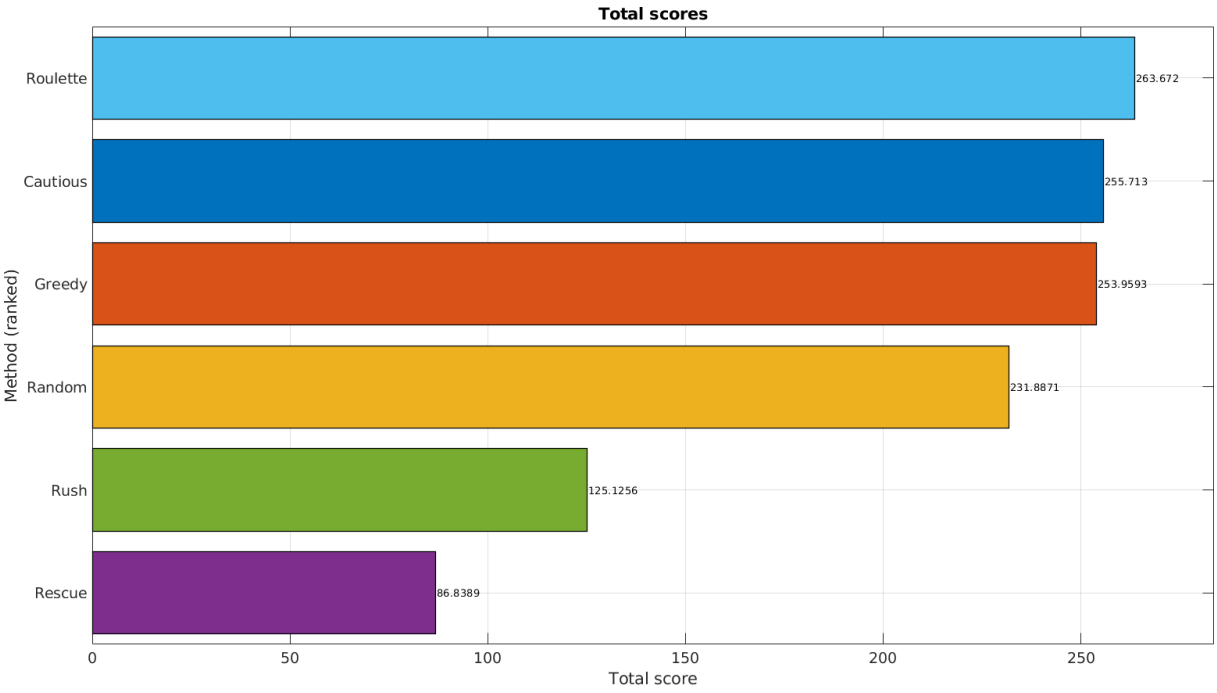


Figure 4.22: Total score ranking for late disturbance scenario.

scenario are visible.

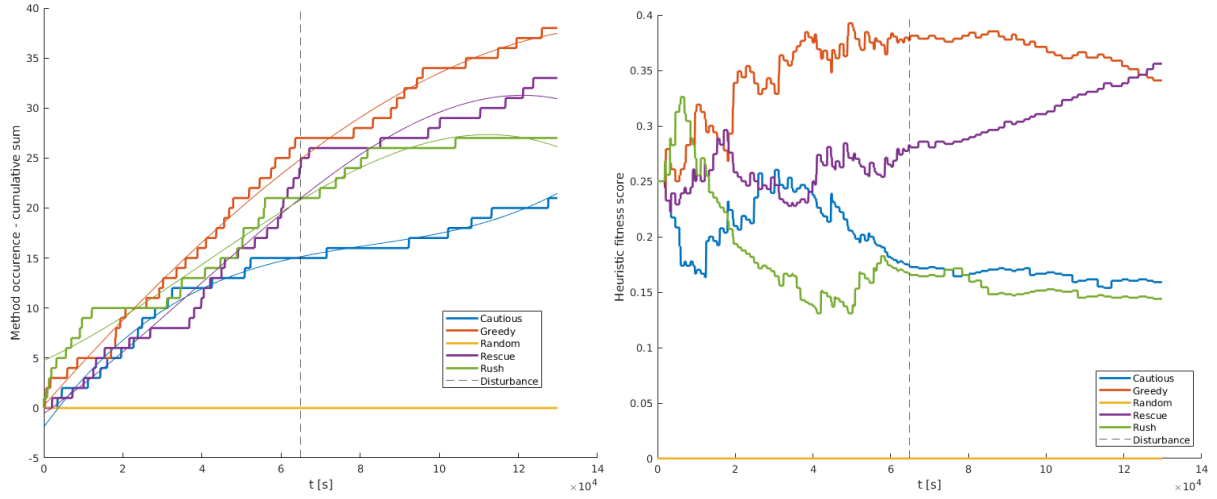


Figure 4.23: Occurrence numbers of each heuristic in the roulette wheel selection (left) and heuristic fitness (right) over time for late disturbance scenario.

4.3.4 Simulation with one outlier aMussel

The aMussels are distributed as shown in 4.24 in a simulated Venice lagoon locale, with one prominent outlier aMussel deployed further away, and operating without disturbance the full length of the mission.

Figure 4.25 shows the values of the performance indices for all heuristics throughout

Figure 4.24: Google Earth image (Image data: ©2022 CNES/Airbus, Maxar Technologies, image acquired 30/07/2022) showing the aMussels in the simulated Venice experiment area. outlier aMussel shown in red.

the length of the experiment. The total achieved aMussel uptime per agent is shown in Figure 4.26. Figure 4.27 shows the final means of achieved scores for all methods, with the best- and worst- performing instances highlighted. Figure 4.28 shows a comparison of differences between scores achieved by each heuristic and the best achieved score, with an overview provided in Table 4.4.

Table 4.4: Scores and ranking - Venice outlier simulation.

Method	Median [%]	Min [%]	Max [%]	Average Rank	Total score
Cautious	30.2912	7.9755	305.3553	3	208.84
Greedy	36.0013	0	83.7288	2.5	230.58
Random	34.0845	8.8779	254.4713	3.5	182.02
Rescue	95.6695	0	370.0055	6	103.00
Rush	92.6978	15.9863	115.8786	4.5	94.44
Roulette	11.6744	0	204.1342	2.5	251.33

Several interesting aspects of the outlier's effect on system behaviour are immediately visible. The impact of Greedy and Rush methods never including the outlier aMussel in their charging schedules is clear, both in the very low movement costs they exhibit, but also by the outlier preservation score being extremely low. Since the outlier is only one

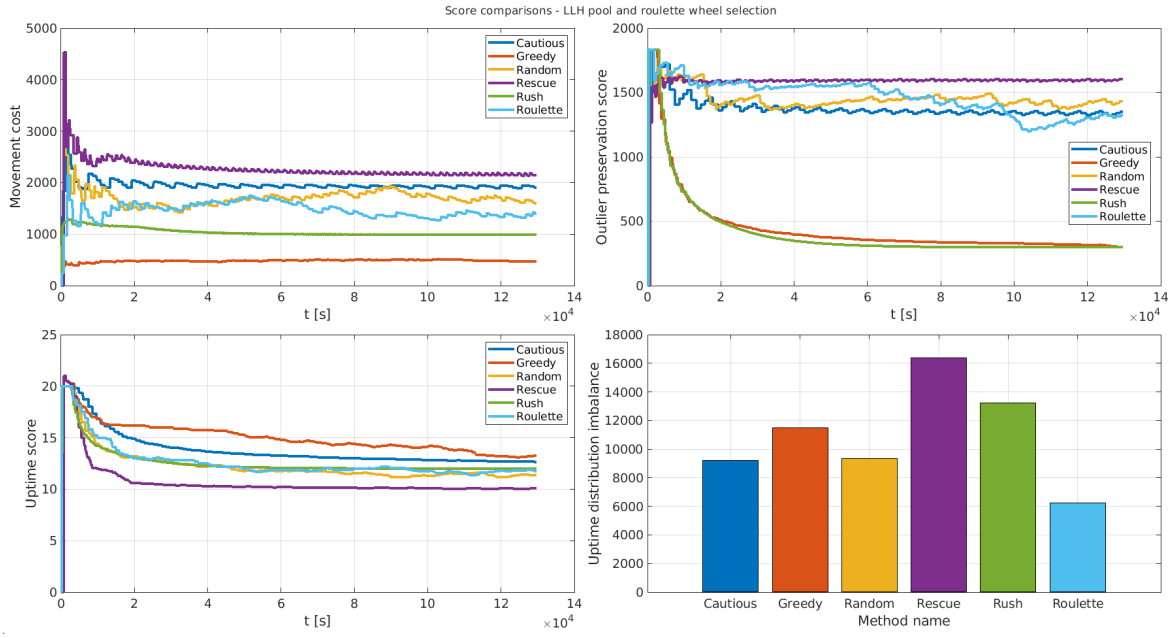


Figure 4.25: Comparison of scores during experimental runs in outlier scenario using only one heuristic, versus roulette wheel selection and random selection.

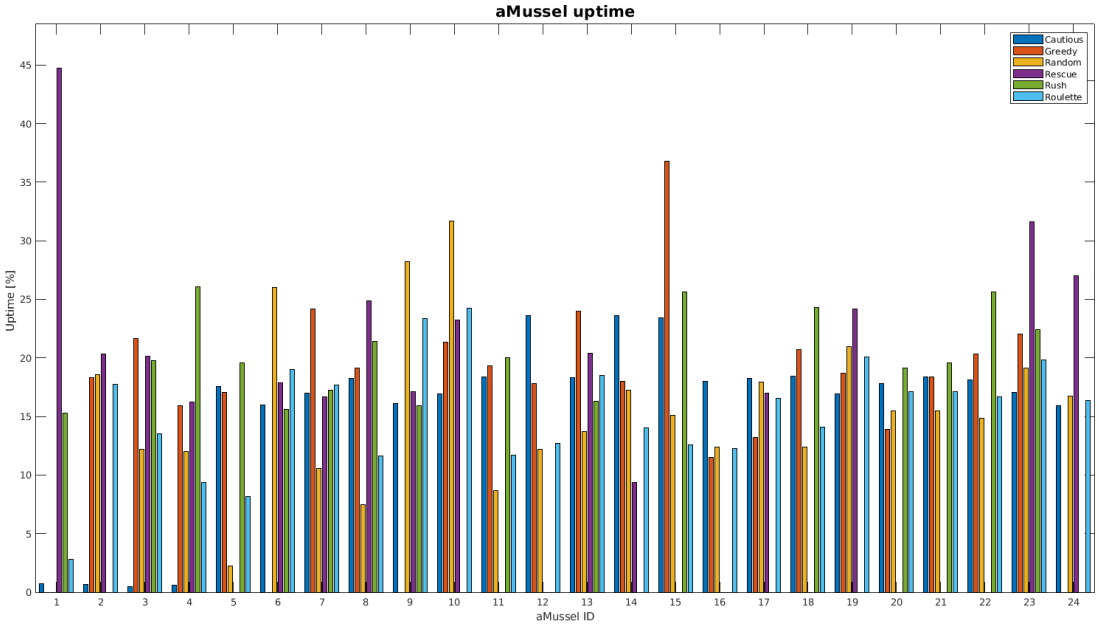


Figure 4.26: Distribution of aMussel uptime in outlier scenario by agent ID in each of the experimental runs.

agent, the overall uptime score is not severely negatively impacted in the case of these two heuristics, however Rescue shows the lowest total uptime score, as it seems to "sacrifice" the uptime of other, closer, aMussels, in order to travel far and keep the outlier charged. Figure 4.29 shows a spider chart breakdown of scores achieved by each heuristic and selection method, with Figure 4.30 showing the ranking of total scores in descending

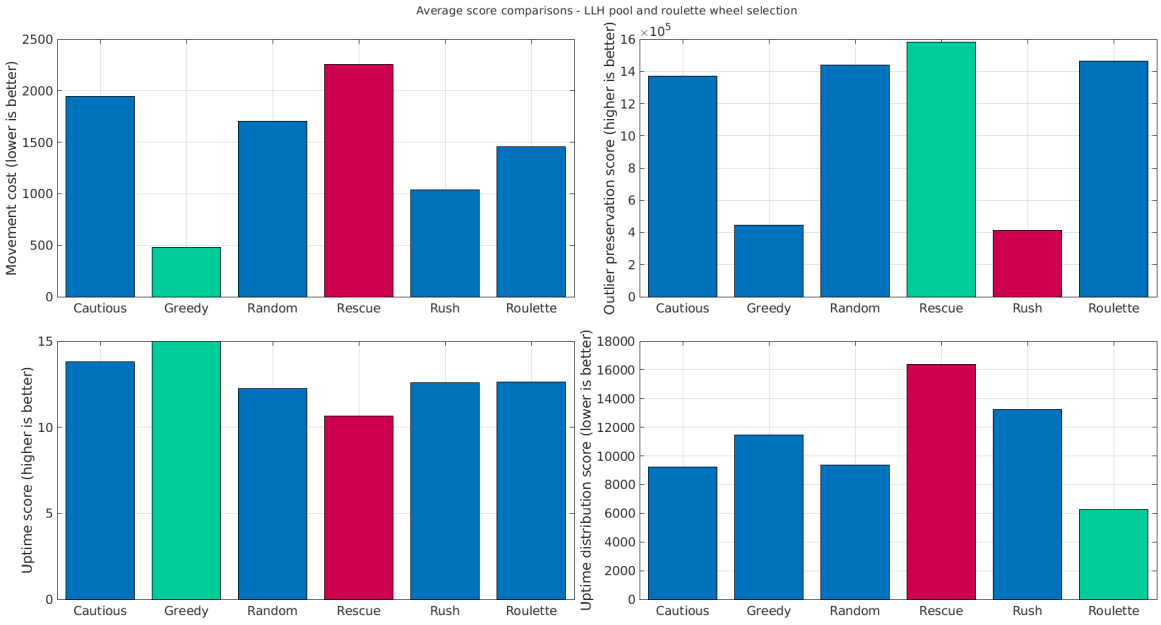


Figure 4.27: Comparison of means of scores achieved by each heuristic in outlier scenario using only one heuristic, versus roulette wheel selection and random selection. Worst-performing heuristic shown in red, best-performing shown in green.

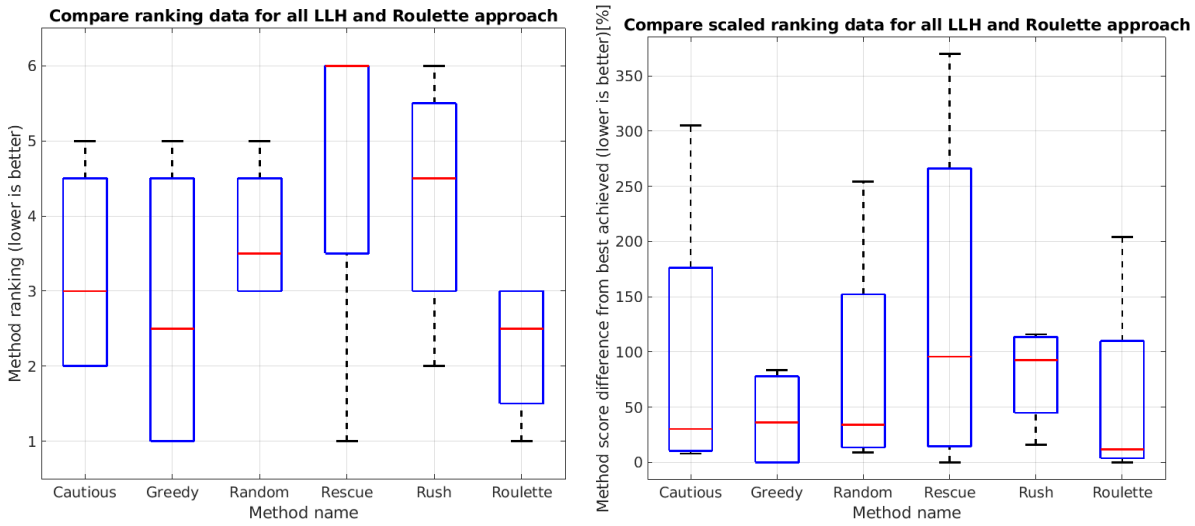


Figure 4.28: Comparison of ranks (left) and differential percentage ranks (right) achieved by each heuristic and roulette wheel selection in outlier scenario.

order.

Figure 4.31 shows when each low-level heuristic was chosen and applied by the roulette wheel selection hyper-heuristic during the scenario, as well as how the fitness values for each method changed. This specific scenario demonstrates the value in combining different heuristics and having their various strengths and weaknesses make up for each other in order to achieve better performance overall and adapt to a specific deployment situation, even without any dramatic changes occurring within the mission duration.

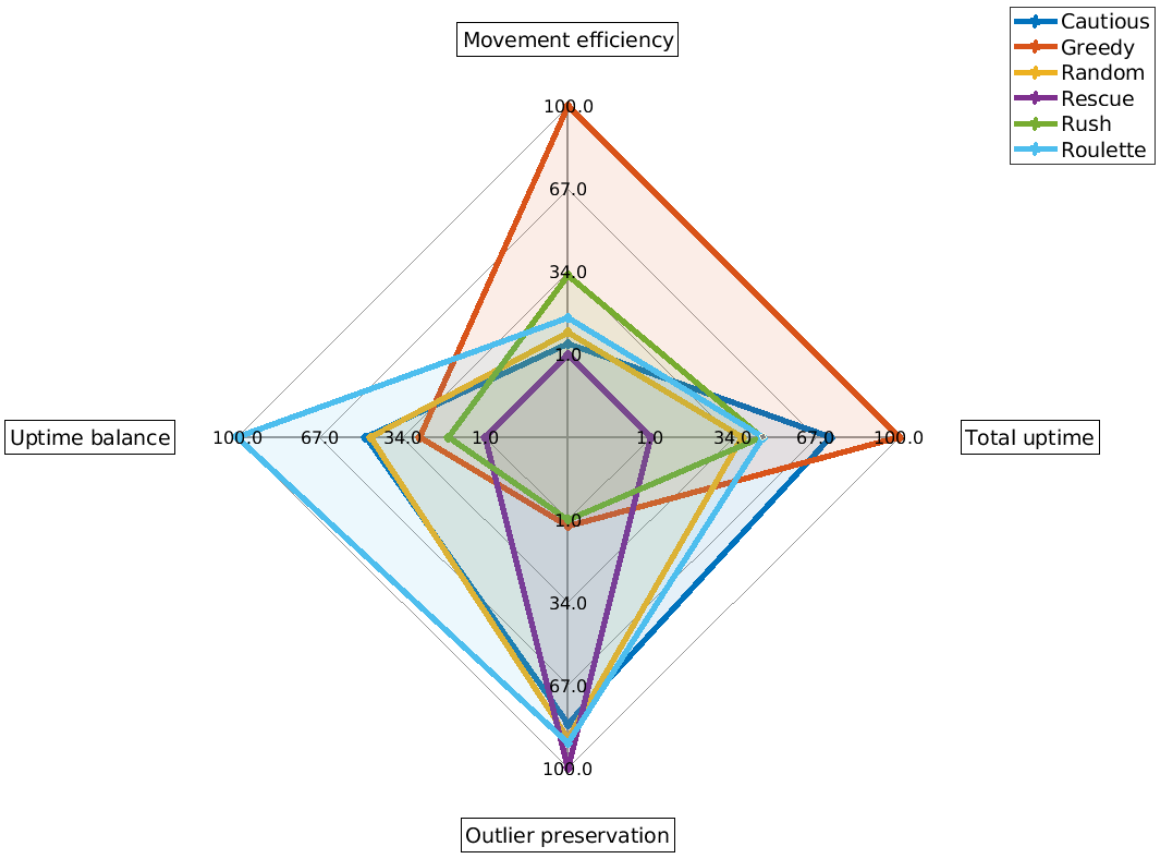


Figure 4.29: Comparison of total scores achieved in the outlier scenario.

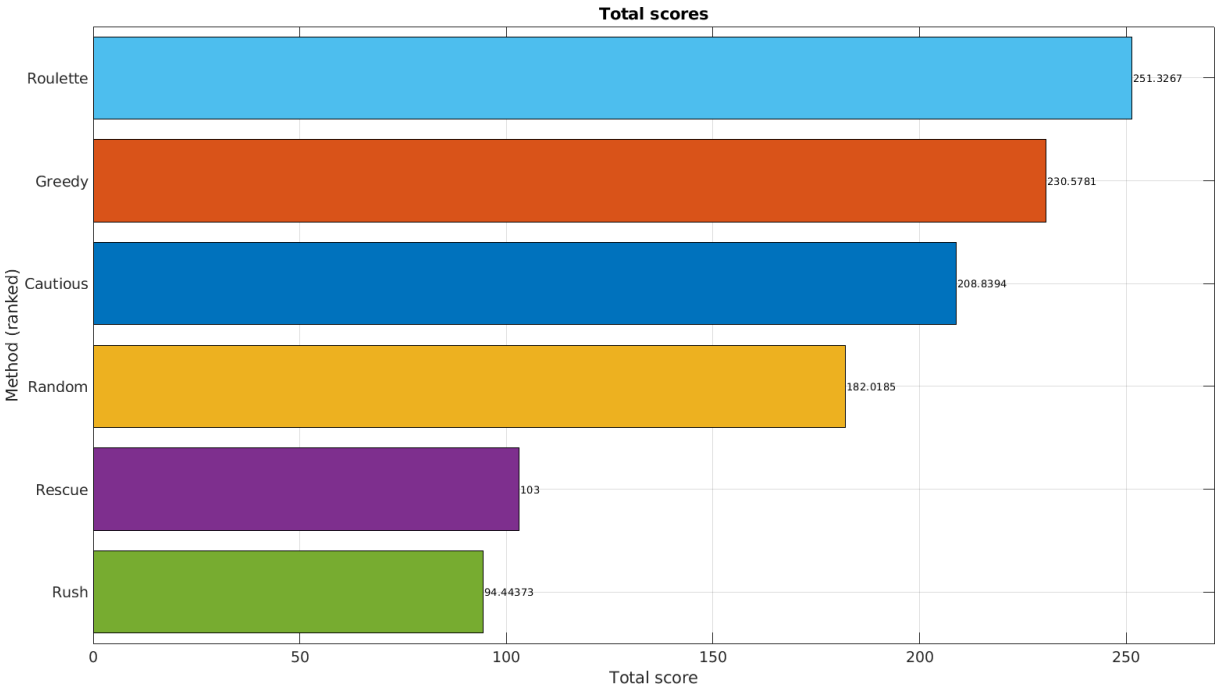


Figure 4.30: Total score ranking for outlier scenario.

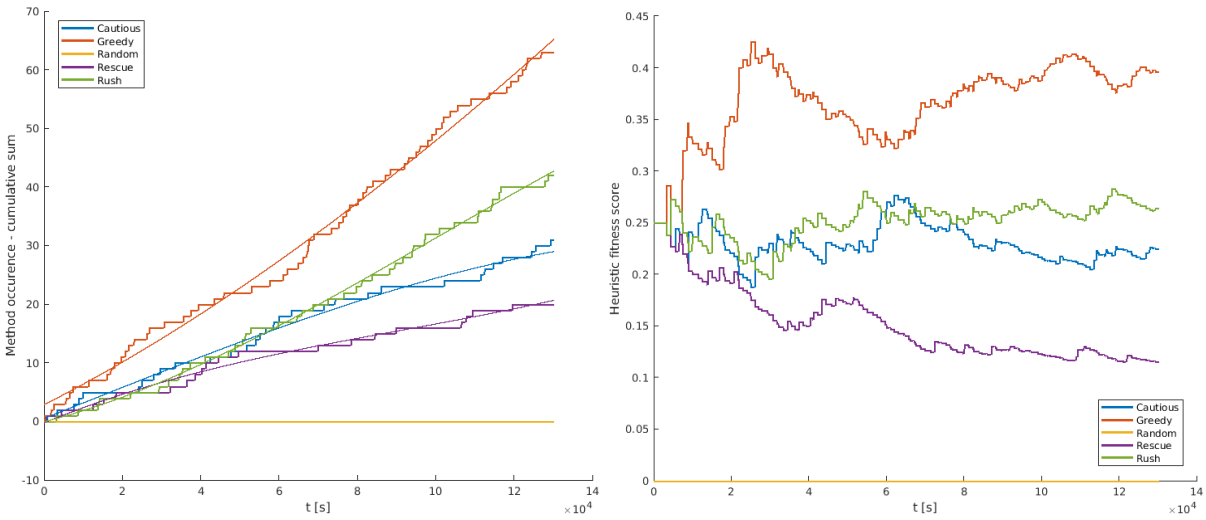


Figure 4.31: Occurrence numbers of each heuristic in the roulette wheel selection (left) and heuristic fitness (right) over time for outlier scenario.

A variety of simulated scenarios successfully demonstrated different behaviours the system is capable of and the adaptability of the hyper-heuristic decision-making to different swarm states and changes over time.

4.4 Hyper-heuristic vehicle-in-the-loop experiments

4.4.1 Proof-of-concept experimental scenario

In Chapter 3, the general outline and early simulated implementations of the aPad decision-making approach were described, in particular the use of k-means clustering to divide the aMussels among several available aPads. One aPad platform at a time is used in the experiment presented here, under the assumption that basic clustering has already happened and the aPads have reached a consensus on which aMussels belong under whose jurisdiction.

The full energy sharing scenario loop for a single aPad is shown in Figure 4.32 and begins after the aPads have been deployed, reached the vicinity of the experimental area, and achieved clustering consensus, meaning all active aPads have their designated patrol areas defined by IDs of aMussels they are responsible for overseeing and maintaining.

The loop itself consists of the following:

- **Idle wait** - The aPad waits, holding its current position using the dynamic positioning mission primitive, until at least four aMussels are present in the charging pool (meaning their battery levels have fallen under a certain predefined threshold).
- **Route planner (collection)** - The aPad plans a route to collect four aMussels chosen from the candidates in the charging pool using one of several heuristics.

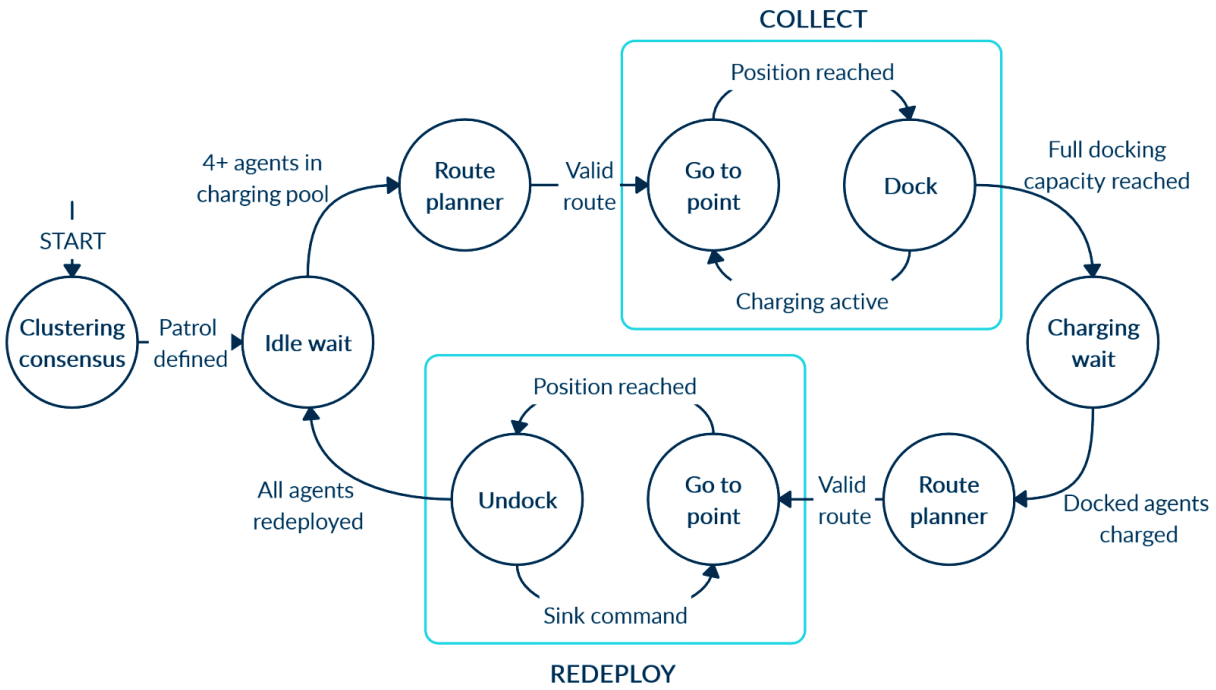


Figure 4.32: Energy exchange scenario loop.

- **Collect** - Four instances of the Go To aMussel position - Dock routine are carried out, until four aMussels have been successfully docked and the aPad is at full docking/charging capacity. Each time the aPad moves to the position of the next aMussel on its collection schedule, it sends it an acoustic message instructing it to surface for collection. This is done so the aMussels spend a minimal amount of time floating on the water surface and also reduces inaccuracies in aMussel position information (as their last localisation is assumed to have happened underwater). For safety reasons, aMussels will never surface without an aPad present in acoustic range giving permission, minimising the risk of losing them to floating away in a current or similar.
- **Charging wait** - The aPad holds position at the location of the last aMussel it picked up, until all four docked and inactive agents report battery levels above 99%.
- **Route planner (deployment)** - The aPad plans a redeployment route using the Greedy redeploy heuristic, bearing in mind all aMussels must be returned to their original positions, starting from the last aMussel it collected.
- **Redeploy** - The aPad alternates between going to the position of the next aMussel to deploy, and undocking (opening the docking mechanism and activating thrusters to back away, freeing the aMussel). Once an aMussel is released, it detects a falling edge on its charging state, and reacts to this by using its buoyancy motors to sink to the bottom of the water and resume collecting measurements - for a simulated

agent, this means transitioning to a discharging battery behaviour and becoming active. The aPad also sends a Wi-Fi data packet with a specific payload to let the aMussel know it can sink without interference.

The scenario can be interrupted by an override from a human operator, or it can be set to end once a certain predefined mission time has been achieved. Another termination condition available is aPad batteries reaching a certain threshold.

The scenario chosen tackles the "worst case scenario" of the aMussel only actively using power from one battery and keeping the other as an emergency backup, while charging assumes the need to charge the slower-charging of the batteries (i.e. using only one charging coil). Concerning the discharge model used, it is assumed there will be no sleep periods allowed during operation, guaranteeing an active aMussel is truly actively performing work in every time step. In normal swarm operation, sleep intervals can be used to spread total aMussel uptime over a longer period (which can be significant for data collection as slow-changing variables are being measured).

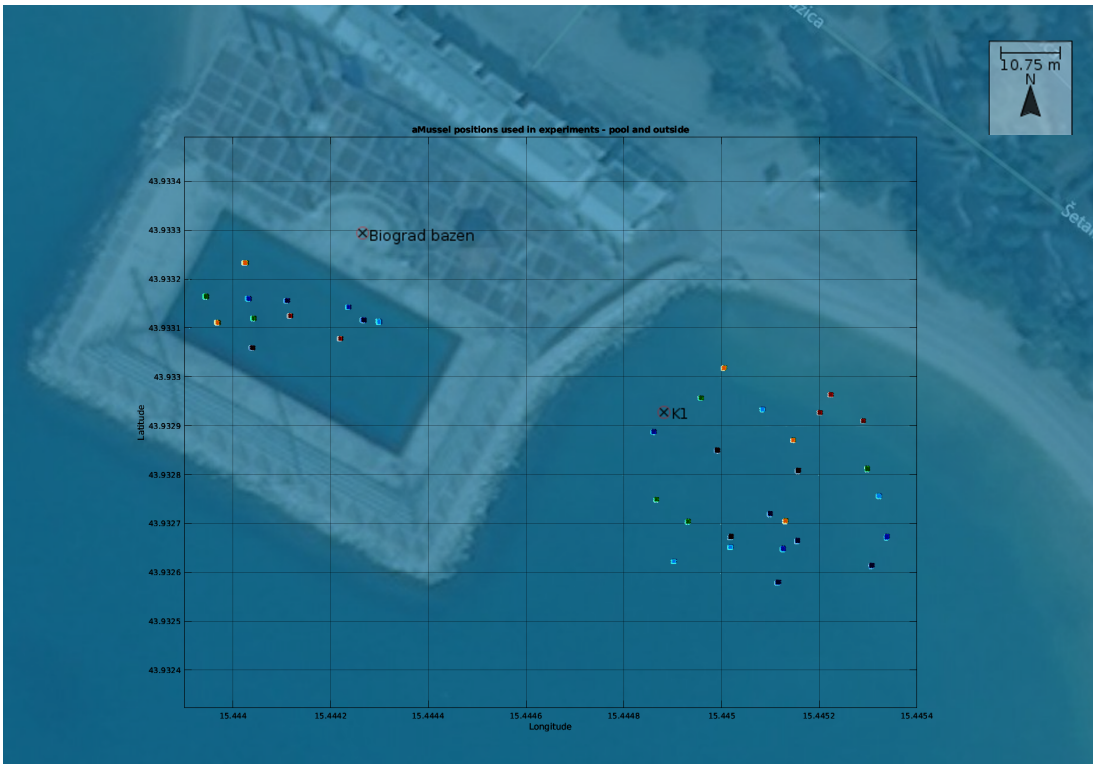


Figure 4.33: The two experimental areas: initial generated aMussel positions used in all experiments, overlaid on map with satellite image.

Either 24 (the actual ratio of aMussel to aPad-type agents in the final subCULTron swarm) or 12 (in the pool experiments due to covering a smaller area) aMussel positions were randomly generated with a uniform distribution within a preset polygon representing the experimental area. Two areas were chosen for initial testing: an outdoor pool (essentially

a walled-off portion of the sea) and a bay next to it that opens up into the Adriatic sea. These generated aMussel positions are shown in Figure 4.33 (located in the same area shown in Figure 3.16). A random drift of $\mathcal{U}(-0.5, 0.5)$ centimetres every second was introduced to simulated aMussel positions to account for them moving and drifting while sinking to or rising from the seabed, or localisation noise. Starting aMussel battery levels were generated from an interval of $[0.6, 1]$. All experiments described were run using the same aMussel configurations for their respective total agent number.

A timescale factor of $\theta = 30$ was used in the experiments, meaning a mission length of 40 minutes represents approximately 20 simulated hours. The node representing and simulating all active aMussels was run on a laptop on the shore connected to the aPad Wi-Fi access point, ensuring realistic communication issues were possible.

The heuristics contrasted in the scenario include collection using the so-called Greedy and Cautious methods, and redeployment using the Greedy Redeploy method.

The vehicle-in-the-loop framework was tested in several experiments in Biograd na Moru, Croatia. Two examples of experiments in progress are shown in Figure 4.34.

Figure 4.34: aPad autonomously carrying out decision-making experiment in pool (left). Experiment outside pool, in nearby bay (right).

Table 4.5 shows a comparison of two experiments done in the pool with 12 mussels and one done outside with 24, as well as a baseline case with no aMussels being charged during the experiment. Each experiment lasted 40 minutes and included four full sets of collections and redeployments. The values being compared are average aMussel uptime, maximum and minimum uptime achieved by any of the aMussels, cumulative movement costs of the aPad, and the average number of aMussels active at a time throughout the experiment.

Figure 4.35 shows an example of the mission replay screen. aMussel locations are displayed, as well as aPad location and a trace of recent trajectory. aMussel markers change colour to indicate status: green for charged above charging pool inclusion threshold, red for charging pool candidates, and purple for currently undergoing charging. Estimated aPad movement cost for each active segment is displayed (replaced by cumulative movement cost at the end of the replay), as well as the currently active mission primitive.

Table 4.5: aMussel uptime, activity, and total aPad movement cost

	Avg %	Max %	Min %	Movement	Active
Cautious	74.37	87.38	59.88	307.1746	8.9243
Greedy (pool)	72.27	87.16	42.84	247.8026	8.9495
Greedy (bay)	71.98	90.31	31.18	186.4318	16.2368
No charging	67.36	79.56	56.64	0	8.0837

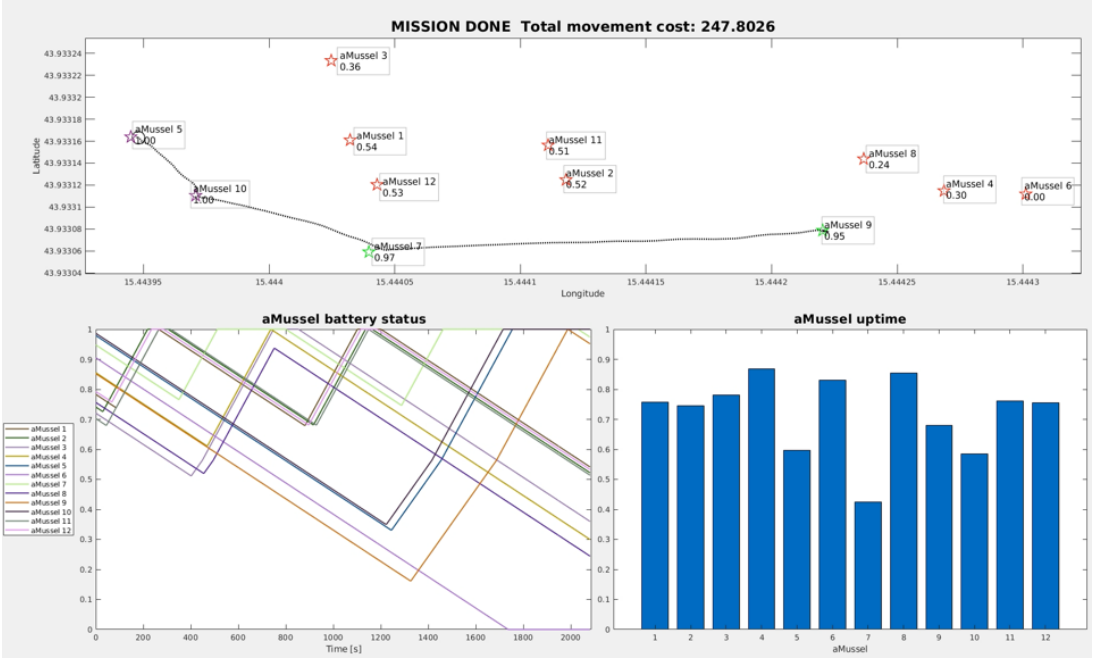


Figure 4.35: Example mission replay screen showing aPad using the Greedy heuristic, working in pool with 12 aMussels.

Battery states of all aMussels are plotted, as is uptime for each agent, updated every simulation step based on known total mission length and current aMussel activity. The number of active aMussels over time is shown in Figure 4.36.

The Greedy method shows less aPad movement overall but more time "wasted" waiting for the lowest battery aMussels to recharge, with some fully charged aMussels needlessly inactive - Cautious has better balancing in that regard, as it makes decisions based on aMussel battery states. Various alternate redeployment solutions are possible to address this, though potentially at the cost of increased aPad movement. aMussel selection with the Greedy method is very affected by the aPad starting position, while with Cautious it is not, beyond the initial movement cost to reach the first docking position.

Due to the fact the aMussels have simulated accelerated time while the aPad doesn't, the aPad seems to move very slowly, relatively (the fairly frequent 30-second bursts of

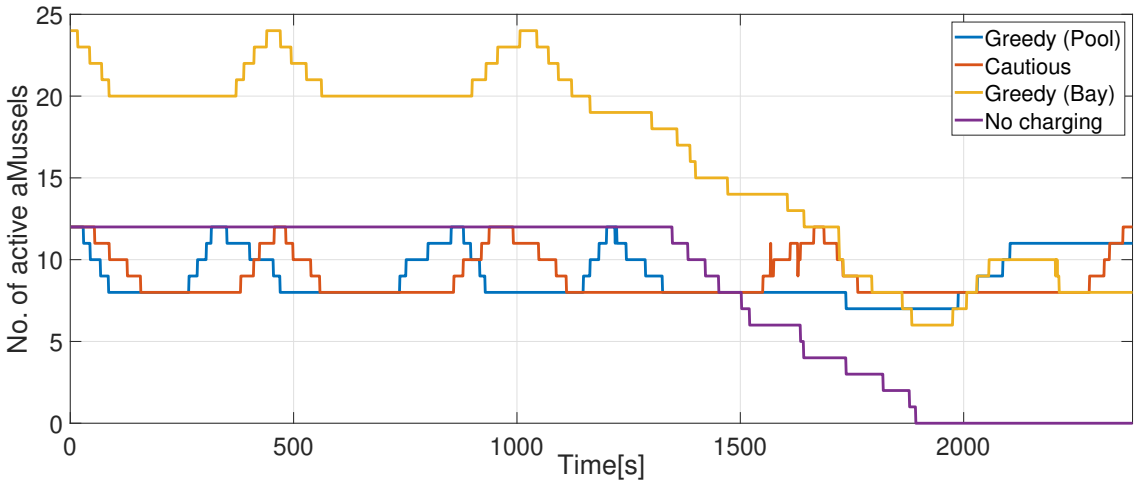


Figure 4.36: Number of currently active aMussels during the experiments.

movement corresponding to 15 minutes in simulated time). This can be interpreted as the aPad travelling over longer distances, however it will of course not be reflected in the actual aPad battery consumption and state. While the mission ended before this could fully come into effect, it can be seen that the aPad is becoming "overwhelmed" by the larger number of aMussels in the example taking place in the bay, leading to the number of active aMussels declining, and highlighting questions of aMussel-per-aPad maintenance capacity. The number of charging cycles a single aPad can feasibly provide is another useful variable to study, for which the developed framework is very helpful, especially in a sense of "rapid prototyping" of behaviours. A video showing two examples of mission replays (one for each collection heuristic) is available at `https://youtu.be/nYiZD0lhF9s`. The developed vehicle-in-the-loop setup can be used to test a variety of specific use-cases of the subCULTron heterogeneous marine robot swarm in a logistically feasible way. The development process resulted in a twofold benefit: collected preliminary data confirmed the long-term operating potential of the aMussel agents themselves, while also providing several different stock behaviour variants for simulation models which will be useful for all future studies of the swarm's agent interactions.

The undertaken proof-of-concept experiments showed variations in aMussels covering two different experimental areas by actively contributing to data collection efforts and required environmental monitoring, while being supported by an aPad platform autonomously engaging in energy sharing using two different heuristic approaches to its task sequencing. These early experiments already provided valuable insights, and reinforced the need for more complex selection and acceptance criteria during the decision-making process to ensure the correct behaviours are rewarded, ultimately leading to long-term sustainable and beneficial swarm behaviours.

4.4.2 Vehicle-in-the-loop experiment with roulette wheel selection

This set of experiments expands upon the initial proof-of-concept VIL setup described above. All runs were performed in the seaside pool in Biograd na Moru using the 12 aMussel pool positions shown in Figure 4.33, with the same parameters and conditions applied with regards to duration, charging pool thresholds, and redeployment. However, here the system used hyper-heuristic roulette wheel selection to pick between a full set of low-level heuristics: Cautious, Greedy, Random, Rescue, and Rush. As was the case with the Venice Lagoon simulations, in order to provide useful data for comparison, experimental runs using only one of each of the low-level heuristics were performed, followed by roulette wheel and random selection runs. The set mission length of 7000s with a timescale of 30 is the equivalent of the swarm operating for 2.5 days.
Figure 4.37 shows the values of the performance indices for all heuristics and both heuristic selection methods throughout the length of the experiment, while the total achieved aMussel uptime per agent is shown in Figure 4.38. Figure 4.39 shows the final means of achieved scores for all methods, with the best- and worst- performing instances highlighted.

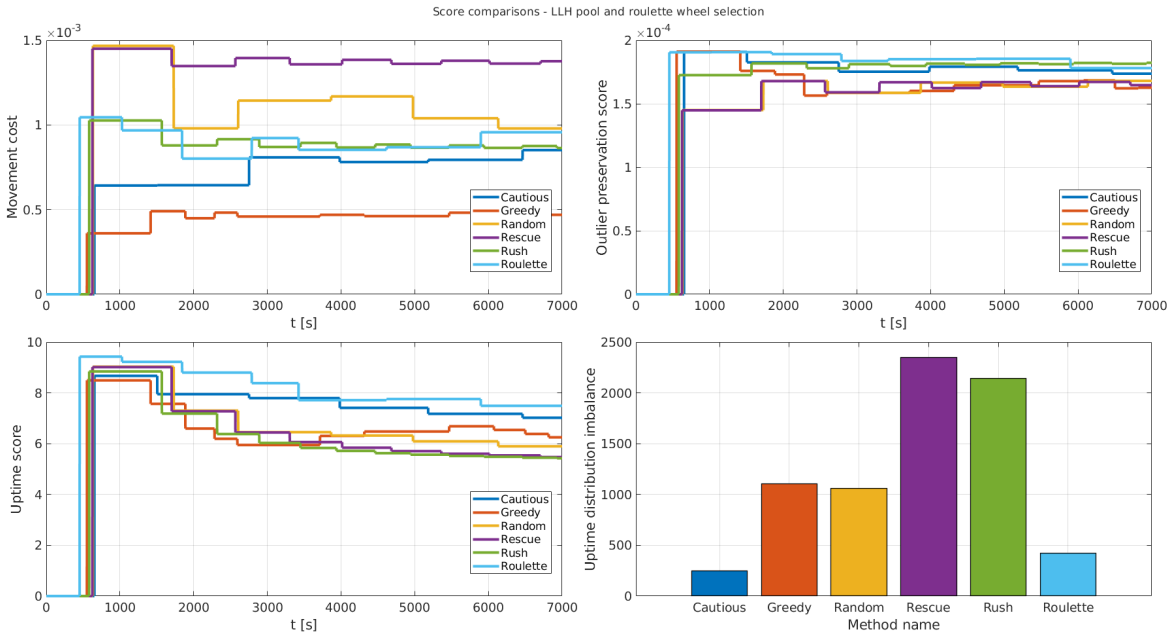


Figure 4.37: Comparison of scores in the VIL experiments using only one heuristic, versus roulette wheel selection and random selection.

Based on the scores shown above, Figure 4.40 shows a comparison of differences between scores achieved by each heuristic and the overall best achieved score, with an overview provided in Table 4.6.

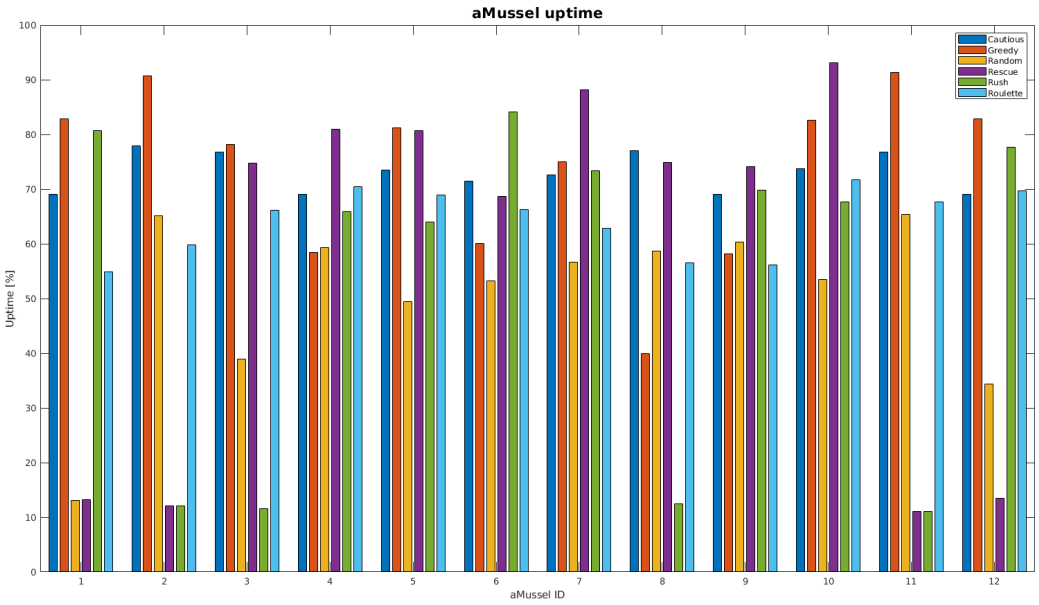


Figure 4.38: Distribution of aMussel uptime in the VIL experiments by agent ID in each of the experimental runs.

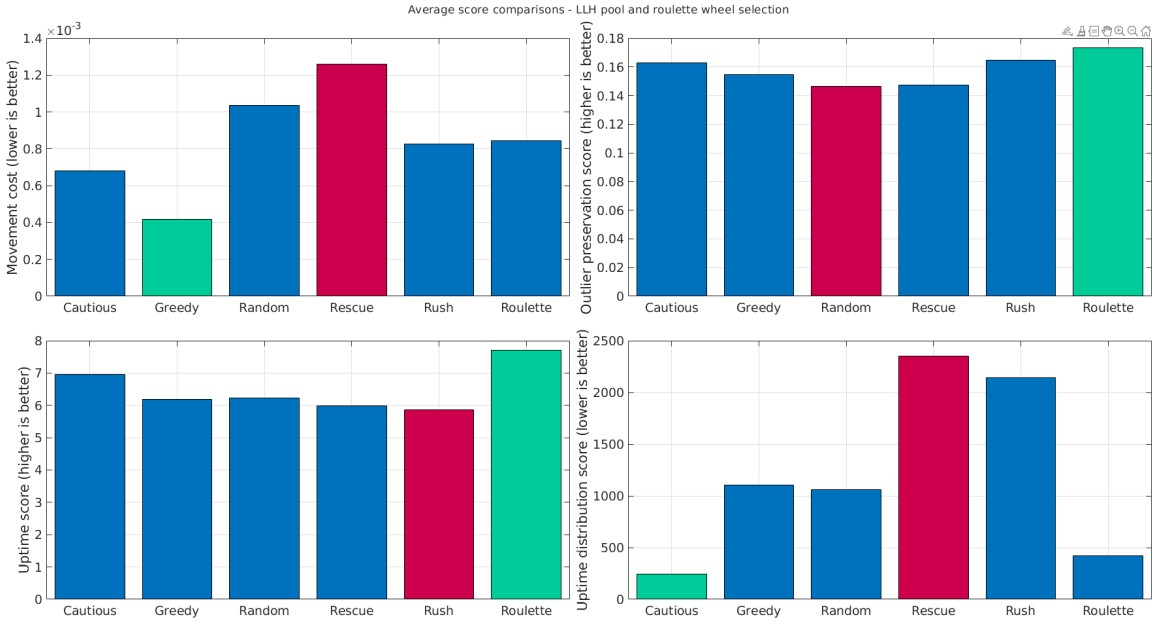


Figure 4.39: Comparison of means of scores achieved by each heuristic in the VIL experiments using only one heuristic, versus roulette wheel selection and random selection. Worst-performing heuristic shown in red, best-performing shown in green.

A spider chart breakdown of total scores achieved by each heuristic as well as the two selection methods is given in Figure 4.41, highlighting the strengths and weaknesses of each approach. A ranking of total scores in descending order are shown in Figure 4.42. A total count of times each low-level heuristic was chosen and applied by the roulette wheel selection hyper-heuristic is shown in Figure 4.43. Figure 4.23 shows how these

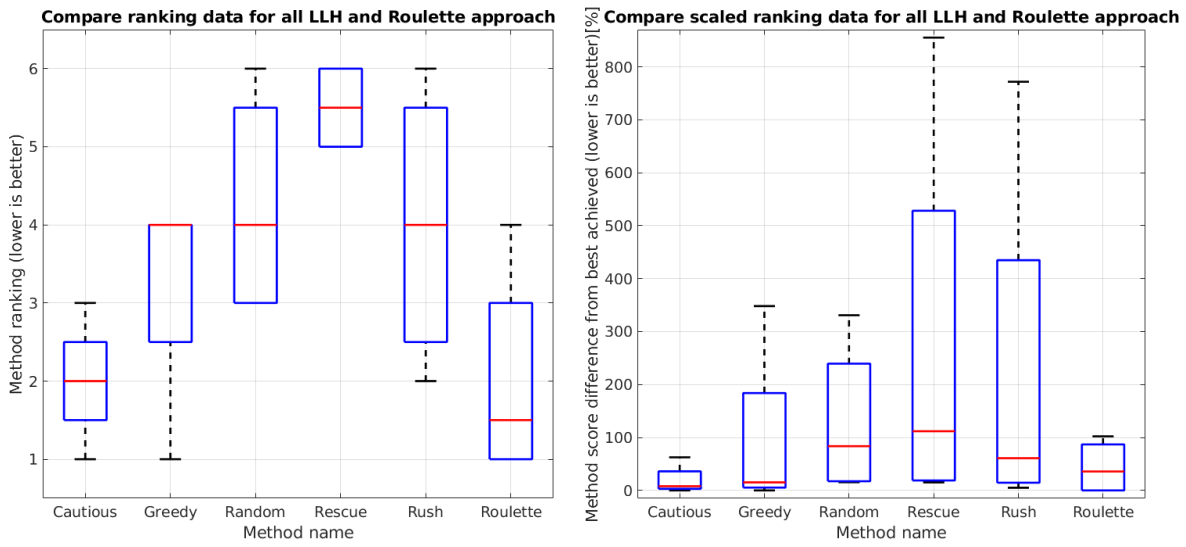


Figure 4.40: Comparison of ranks (left) and differential percentage ranks (right) achieved by each heuristic and roulette wheel selection in the VIL experiments.

Table 4.6: Scores and ranking - Biograd VIL experiment.

Method	Median [%]	Min [%]	Max [%]	Average Rank	Total score
Cautious	7.9364	0	62.5026	2	263.80
Greedy	15.2571	0	348.1128	4	164.14.95
Random	83.5881	15.5706	330.9366	4	48.5354
Rescue	111.7443	15.2543	855.2184	5.5	13.03
Rush	60.9051	5.0895	771.7804	4	97.48
Roulette	35.8229	0	102.1949	1.5	278.93

choices were distributed in the scenario's duration, as well as how the fitness values for each method changed. Due to the relatively short length of the experiments, far fewer selections and evaluations were performed.

A highly pronounced effect of only applying a single heuristic is visible here in the case of Rescue achieving very low scores, especially in outlier preservation, which it is supposed to address. The reason for this is that by constantly focusing on and charging the most distant aMussels and outliers, their active uptime is actually detrimentally impacted. In comparison, using it combined with other heuristics leads to excellent scores, especially in this specific case where a smaller number of aMussels is positioned relatively close together.

A playlist of videos of mission replays of the roulette wheel and single heuristic experiments is available at `https://www.youtube.com/playlist?list=PL9hXWi5RIHgRjql-Tkh_SNQfCoSpwVSsA`.

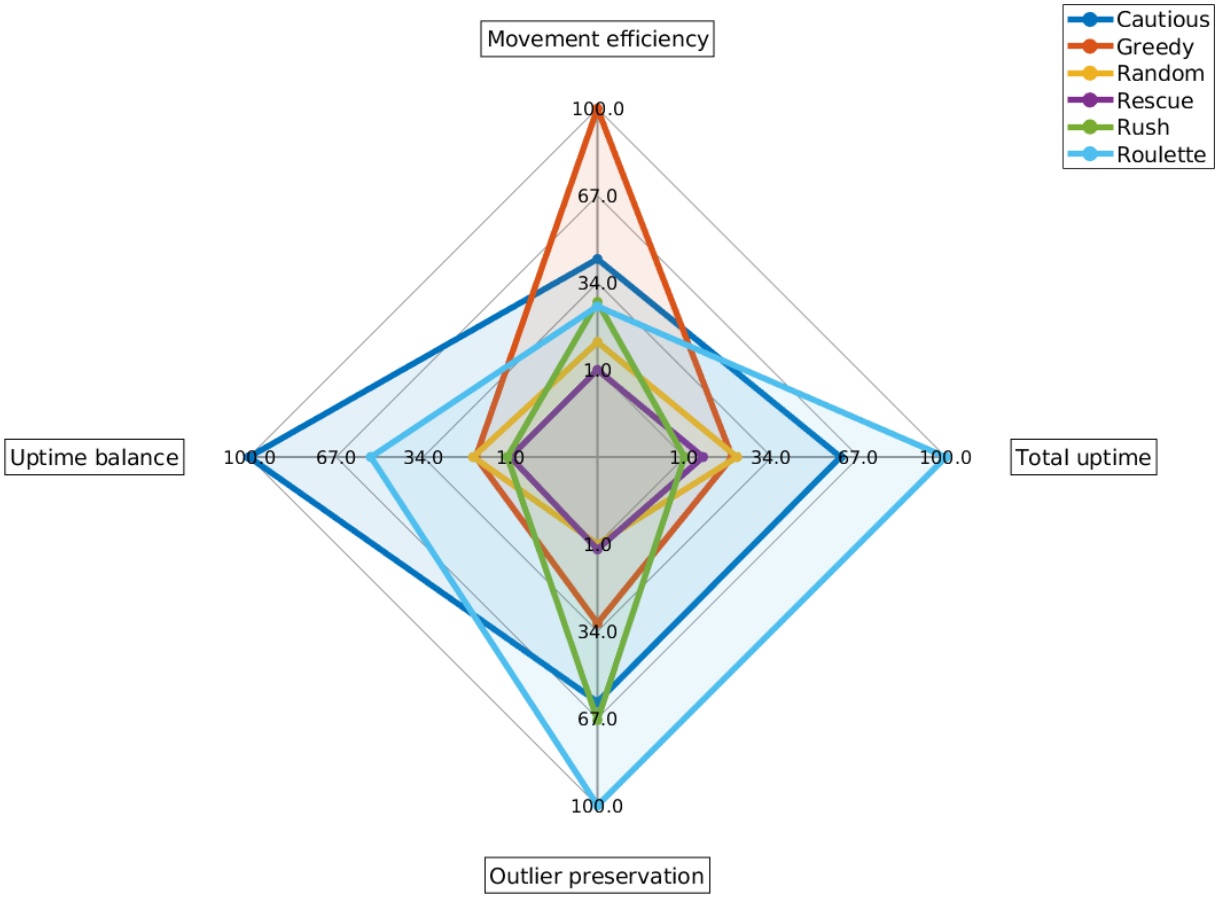


Figure 4.41: Comparison of total scores achieved in the VIL experiments.

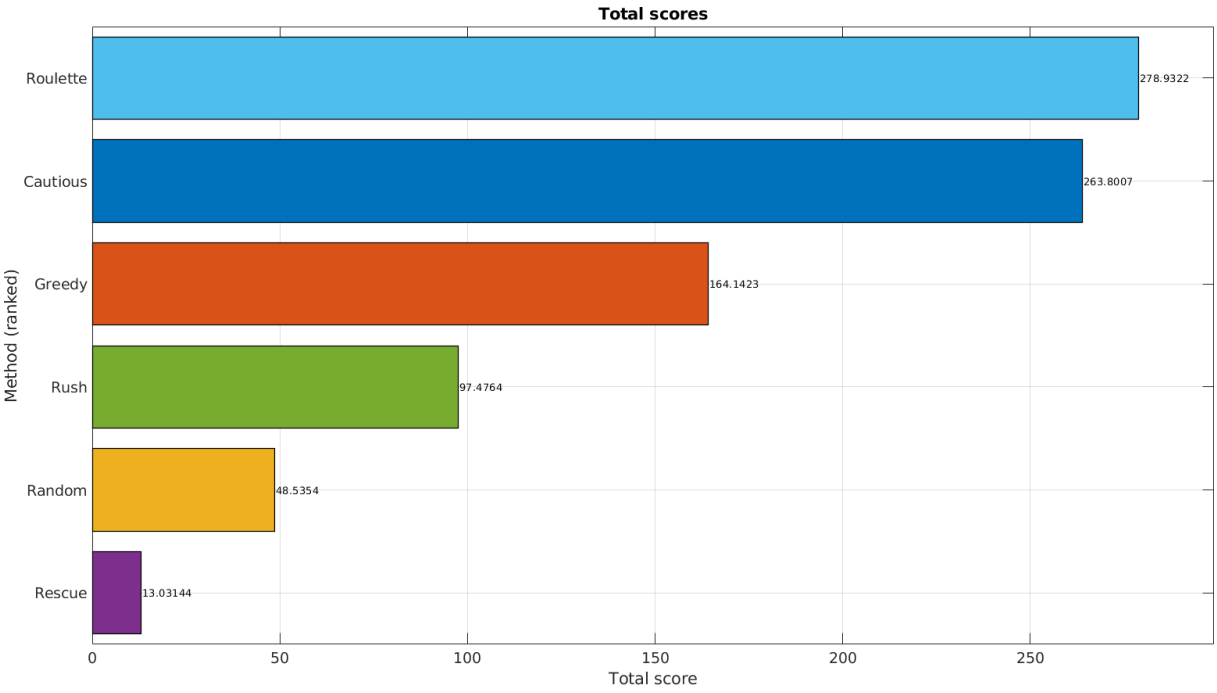


Figure 4.42: Total score ranking for the VIL experiments.

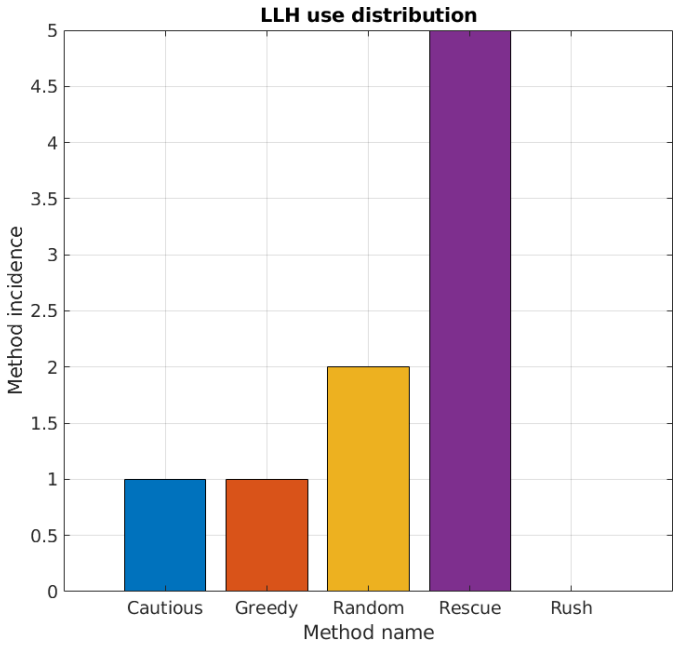


Figure 4.43: Distribution of low-level heuristics selected by roulette wheel in the VIL experiments.

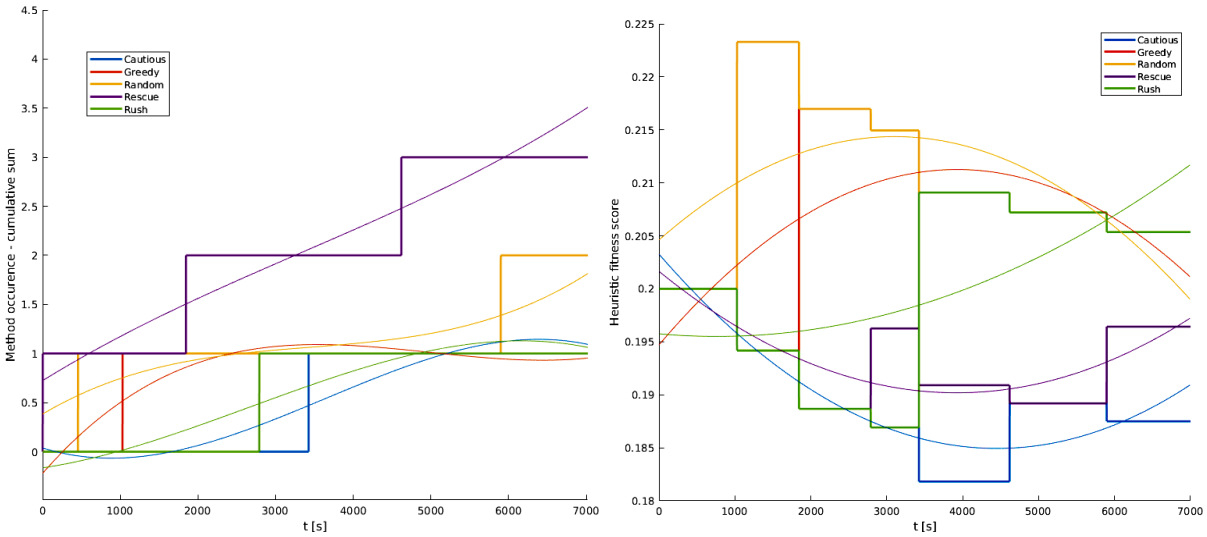


Figure 4.44: Occurrence numbers of each heuristic in the roulette wheel selection (left) and heuristic fitness (right) over time for the VIL experiments.

The results of the simulations and VIL experiments confirm the hypothesis that it is possible to continuously generate sequences of simple and computationally undemanding heuristics that perform better than repeatedly applying each of the individual heuristics, as well as applying them randomly. As a hyper-heuristic structure with a firmly established domain barrier, the system can generalise to various types of instances and novel situations arising in the problem space, e.g. changes in the types, numbers, and capabilities of swarm agents, environmental conditions, and deployment in different environments in general. However, an ideal use-case would be training the hyper-heuristic on

a simulated or recorded instance with similar features to the desired planned deployment configuration, then employing a pre-weighted roulette wheel selection, whilst still keeping the reinforcement learning active and thus enabling further adaptation, if necessary.
The simulated aMussel structure can also be run on the aPads themselves during missions with real sensor nodes, serving as a model for the aPads to estimate aMussel status and fill in any potential information gaps without needing to receive communication packets from the real agents who might be asleep, out of range, or subject to some other cause of interference and data loss. This way, aPads can pre-emptively make decisions and plan their missions, adjusting to incoming data in real time as they work.

Chapter 5

Conclusion

For the purpose of enabling long-term autonomy of a heterogeneous swarm of marine robots, task allocation and sequencing were introduced into the system's energy management procedures and agent interactions. In a scenario where the system needs to autonomously go about its monitoring mission and survive long-term, the available maximum capacity of five surface vehicles - aPad platforms which represent the charging hubs of the system - is usually outnumbered by the number of active charging requests by the sensor node-like aMussel agents, leading to a need for careful planning and optimisation of robot activities. In the scope of this thesis, a two-layered system of decision-making algorithms was developed: a low-level specific solution-focused set of algorithms, and a high-level hyper-heuristic which selects between them, evaluates performance achieved in each step of the monitoring mission, and employs reinforcement learning to enable a level of adaptation to unknown environments, environmental changes, or changes to the agents themselves. The thesis stated several hypotheses and three major contributions in Section 1.1. These contributions are restated and reviewed in the context of the presented work.

The first contribution stated:

- *A method for multi-robot task assignment and sequencing ensuring long-term autonomy of a heterogeneous swarm of marine robots while taking into account environmental constraints.*

In Chapter 3 a detailed description was given of the structure of the decision-making system applied to the heterogeneous swarm of marine robots, with an overview of the studied problem of the long-term monitoring mission. Methods of initial patrol zone assignment and task partitioning, including a combination of differential evolution and k-means clustering, along with the influence of water currents as a primary example of environmental constraints, were described in 3.3. Initial discrete event-based simulations of a multi-robot system were given in Section 3.5, whereas experimental validation of

the realised task assignment system, demonstrating full communication, decision-making, and consensus-reaching abilities between several robotic agents, was shown in Section 3.6. Examples of the system reacting to and adapting to a disturbance were given in Sections 4.3.2 and 4.3.3, while Section4.3.4 showed how the system adapted to a specific experimental area and agent distribution featuring a geographical outlier.
The second contribution stated:

- *A hyper-heuristic decision-making method for a heterogeneous swarm of marine robots based on unsupervised switching between multiple task assignment and sequencing methods.*

The framework developed for a hyper-heuristic decision-making structure was given in detail in Chapter 4. The method described in this thesis featured selecting between an array of situational low-level heuristic methods of task sequencing described in Section 3.4. It also employed online reinforcement learning based on rewarding or punishing certain methods to encourage or discourage their use, using performance indices for solution evaluation after each decision-making and energy exchange cycle performed by the agents. This enabled fully autonomous and unsupervised functioning of the heterogeneous robotic swarm.
The third contribution stated:

- *A solution evaluation method and definition of performance indices and a benchmark validation scenario for decision-making and task assignment methods implemented on a heterogeneous swarm of marine robots.*

Chapter 2 presented the heterogeneous swarm of marine robots and both hardware and software implementation details related to achieving behaviours enabling the success of a long-term environmental monitoring mission, from autonomous docking and energy exchange between swarm agents, to a VIL setup which made prototyping and testing on real vehicles viable. Individual solution performance evaluation for the purpose of adjusting fitness during the learning process spanning the length of every scenario was based on the four indices presented in Section 3.4, while a method of ranking, scoring, and benchmarking the performance of various decision-making methods was given in Section 4.2. The defined benchmark validation was then applied to a series of simulated scenarios. These scenarios and their results showing the successful use of autonomous decision-making and task assignment methods in a variety of situations were described in Section 4.3, with experimental validation given in Section 4.4.

Bibliography

[1]Tagliapietra, D., Aloui Bejaoui, N., Bellafiore, D., De Wit, R., Ferrarin, C., Gamito, S., Lasserre, P., Magni, P., Mistri, M., Pérez Ruzafa, A. *et al.*, "The ecological implications of climate change on the Lagoon of Venice", UNESCO Digital Library, 2011.

[2]Ferrighi, A., Flooding and environmental challenges for Venice and its lagoon: state of knowledge. Cambridge University Press, 2005.

[3]Sorokin, Y. I., Sorokin, P. Y., Giovanardi, O., Dalla Venezia, L., "Study of the ecosystem of the lagoon of Venice, with emphasis on anthropogenic impact", Marine Ecology Progress Series, Vol. 141, No. 1-3, 1996, pages 247–261, available at: 10.3354/meps141247

[4]Argese, E., Cogoni, G., Zaggia, L., Zonta, R., Pini, R., "Study on redox state and grain size of sediments in a mud flat of the Venice Lagoon", Environmental Geology and Water Sciences, Vol. 20, No. 1, jul 1992, pages 35–42, available at: 10.1007/BF01736108

[5]Solidoro, C., Pecenik, G., Pastres, R., Franco, D., Dejak, C., "Modelling macroalgae (Ulva rigida) in the Venice lagoon: Model structure identification and first parameters estimation", Ecological Modelling, Vol. 94, No. 2-3, jan 1997, pages 191–206, available at: 10.1016/S0304-3800(96)00025-7

[6]Lang, F., von der Lippe, M., Schimpel, S., Scozzafava-Jaeger, T., Straub, W., "Topsoil morphology indicates bio-effective redox conditions in Venice salt marshes", Estuarine, Coastal and Shelf Science, Vol. 87, No. 1, mar 2010, pages 11–20, available at: 10.1016/j.ecss.2009.12.002

[7]Thenius, R., Moser, D., Varughese, J. C., Kernbach, S., Kuksin, I., Kernbach, O., Kuksina, E., Mišković, N., Bogdan, S., Petrović, T., Babić, A., Boyer, F., Lebastard, V., Bazeille, S., Ferrari, G. W., Donati, E., Pelliccia, R., Romano, D., van Vuuren, G. J., Stefanini, C., Morgantin, M., Campo, A., Schmickl, T., "subCULTron - Cultural development as a tool in underwater robotics", in Communications in Computer and Information Science, Vol. 732, 2018, pages 27–41, available at: 10.1007/978-3-319-90418-4_3

[8]Tuna, G., Gungor, V. C., "A survey on deployment techniques, localization algorithms, and research challenges for underwater acoustic sensor networks", International Journal of Communication Systems, Vol. 30, No. 17, nov 2017, page e3350, available at: 10.1002/dac.3350

[9]Aguilar, W., Santamaría-Bonfil, G., Froese, T., Gershenson, C., "The past, present, and future of artificial life", Frontiers in Robotics and AI, Vol. 1, oct 2014, page 8, available at: 10.3389/frobt.2014.00008

[10]Canese, L., Cardarilli, G. C., Di Nunzio, L., Fazzolari, R., Giardino, D., Re, M., Spanò, S., "Multi-agent reinforcement learning: A review of challenges and

applications", available at: 10.3390/app11114948 2021.

[11]Brooks, R. A., "Intelligence without representation", Artificial Intelligence, Vol. 47, No. 1-3, jan 1991, pages 139–159, available at: 10.1016/0004-3702(91)90053-M

[12]Mouret, J. B., Doncieux, S., "Encouraging behavioral diversity in evolutionary robotics: An empirical study", Evolutionary Computation, Vol. 20, No. 1, 2012, pages 91–133, available at: 10.1162/EVCO_a_00048

[13]Burtsev, M., Turchin, P., "Evolution of cooperative strategies from first principles", Nature, Vol. 440, No. 7087, 2006, pages 1041–1044, available at: 10.1038/nature04470

[14]Trianni, V., Tuci, E., Passino, K. M., Marshall, J. A., "Swarm Cognition: An interdisciplinary approach to the study of self-organising biological collectives", Swarm Intelligence, Vol. 5, No. 1, 2011, pages 3–18, available at: 10.1007/s11721-010-0050-8

[15]Parker, L. E., "Distributed Intelligence: Overview of the Field and its Application in Multi-Robot Systems", Journal of Physical Agents, Vol. 2, No. 1, 2008, pages 5–14, available at: 10.14198/JoPha.2008.2.1.02

[16]Pagello, E., D'Angelo, A., Ferrari, C., Polesel, R., Rosati, R., Speranzon, A., "Emergent behaviors of a robot team performing cooperative tasks", Advanced Robotics, Vol. 17, No. 1, jan 2003, pages 3–19, available at: 10.1163/156855303321125596

[17]Haasdijk, E., Bredeche, N., Eiben, A. E., "Combining environment-driven adaptation and task-driven optimisation in evolutionary robotics", PLoS ONE, Vol. 9, No. 6, jun 2014, page e98466, available at: 10.1371/journal.pone.0098466

[18]Trueba, P., Prieto, A., Bellas, F., "Distributed embodied evolution for collective tasks: Parametric analysis of a canonical algorithm", in GECCO 2013 - Proceedings of the 2013 Genetic and Evolutionary Computation Conference Companion. New York, NY, USA: ACM, jul 2013, pages 37–38, available at: 10.1145/2464576.2464595

[19]Ficici, S. G., Watson, R. A., Pollack, J. B., "Embodied Evolution: A Response to Challenges in Evolutionary Robotics", Proceedings of the Eighth European Workshop on Learning Robots, 1999, pages 14–22, available at: www.demo.cs.brandeis.edu

[20]Teacy, W. T., Chalkiadakis, G., Farinelli, A., Rogers, A., Jennings, N. R., McClean, S., Parr, G., "Decentralized Bayesian reinforcement learning for online agent collaboration", in 11th International Conference on Autonomous Agents and Multiagent Systems 2012, AAMAS 2012: Innovative Applications Track, Vol. 1, 2012, pages 312–319, available at: https://www.semanticscholar.org/paper/Decentralized-Bayesian-reinforcement-learning-for-Teacy-Chalkiadakis/32ca77bfcb87d7894f5601590ea49bc7d9b07122

[21]Weiss, G., Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, 1999, Vol. 3, No. 2, available at: http://books.google.com/books?hl=nl&lr=&id=JYcznFCN3xcC&pgis=1

[22]Buşoniu, L., Babuška, R., De Schutter, B., "A comprehensive survey of multiagent reinforcement learning", pages 156–172, available at: 10.1109/TSMCC.2007.913919 mar 2008.

[23]Kok, J. R., Vlassis, N., "Collaborative multiagent reinforcement learning by payoff propagation", Journal of Machine Learning Research, Vol. 7, 2006, pages 1789–1828,

available at: https://www.jmlr.org/papers/volume7/kok06a/kok06a.pdf

[24]Uchibe, E., Doya, K., "Finding intrinsic rewards by embodied evolution and constrained reinforcement learning", Neural Networks, Vol. 21, No. 10, dec 2008, pages 1447–1455, available at: 10.1016/j.neunet.2008.09.013

[25]Schembri, M., Mirolli, M., Baldassarre, G., "Evolving internal reinforcers for an intrinsically motivated reinforcement-learning robot", in 2007 IEEE 6th International Conference on Development and Learning, ICDL. IEEE, jul 2007, pages 282–287, available at: 10.1109/DEVLRN.2007.4354052

[26]Singh, S., Barto, A. G., Chentanez, N., "Intrinsically motivated reinforcement learning", in Advances in Neural Information Processing Systems, 2005, available at: https://proceedings.neurips.cc/paper/2004/file/4be5a36cbaca8ab9d2066debfe4e65c1-Paper.pdf

[27]Elfwing, S., Uchibe, E., Doya, K., Christensen, H. I., "Darwinian embodied evolution of the learning ability for survival", Adaptive Behavior, Vol. 19, No. 2, apr 2011, pages 101–120, available at: 10.1177/1059712310397633

[28]Bredeche, N., Montanier, J. M., Liu, W., Winfield, A. F., "Environment-driven distributed evolutionary adaptation in a population of autonomous robotic agents", Mathematical and Computer Modelling of Dynamical Systems, Vol. 18, No. 1, feb 2012, pages 101–129, available at: 10.1080/13873954.2011.601425

[29]Jakobi, N., Husbands, P., Harvey, I., "Noise and the reality gap: The use of simulation in evolutionary robotics", in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 929, 1995, pages 704–720, available at: 10.1007/3-540-59496-5_337

[30]Zagal, J. C., Ruiz-Del-Solar, J., Vallejos, P., "Back to reality: Crossing the reality gap in evolutionary robotics", in IFAC Proceedings Volumes (IFAC-PapersOnline), Vol. 37, No. 8. Elsevier, jul 2004, pages 834–839, available at: 10.1016/s1474-6670(17)32084-0

[31]Billing, E. A., "Cognitive perspectives on robot behavior", in ICAART 2010 - 2nd International Conference on Agents and Artificial Intelligence, Proceedings, Vol. 2, 2010, pages 373–382, available at: 10.5220/0002782103730382

[32]Kurabayashi, D., "Toward realization of collective intelligence and emergent robotics", in Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Vol. 4. IEEE, 1999, pages 748–753, available at: 10.1109/icsmc.1999.812498

[33]Babić, A., Lončar, I., Mišković, N., Vukić, Z., "Energy-efficient environmentally adaptive consensus-based formation control with collision avoidance for multi-vehicle systems", IFAC-PapersOnLine, Vol. 49, No. 23, jan 2016, pages 361–366, available at: 10.1016/j.ifacol.2016.10.431

[34]Schlimmer, J. C., Langley, P., "Paradigms for machine learning", Vol. 40, 1991, pages 1–9, available at: https://ntrs.nasa.gov/search.jsp?R=19920016857http://hdl.handle.net/2060/19920016857

[35]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, apr 2019, page 1388, available at: 10.3390/app9071388

[36]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 (Switzerland), Vol. 20, No. 16, aug 2020, pages 1–23, available at: 10.3390/s20164615

[37]Akyildiz, I. F., Pompili, D., Melodia, T., “Underwater acoustic sensor networks: research challenges”, Ad Hoc Networks, Vol. 3, No. 3, 2005, pages 257–279, available at: https://doi.org/10.1016/j.adhoc.2005.01.004

[38]Akyildiz, I. F., Pompili, D., Melodia, T., “State-of-the-art in protocol research for underwater acoustic sensor networks”, in Proceedings of the 1st ACM International Workshop on Underwater Networks, ser. WUWNet ’06. New York, NY, USA: ACM, 2006, pages 7–16, available at: 10.1145/1161039.1161043

[39]Tan, H.-P., Diamant, R., Seah, W. K., Waldmeyer, M., “A survey of techniques and challenges in underwater localization”, Ocean Engineering, Vol. 38, No. 14, 2011, pages 1663–1676, available at: https://doi.org/10.1016/j.oceaneng.2011.07.017

[40]Tuna, G., Gungor, V. C., “A survey on deployment techniques, localization algorithms, and research challenges for underwater acoustic sensor networks”, International Journal of Communication Systems, Vol. 30, No. 17, page e3350, e3350 IJCS-16-0556.R1, available at: 10.1002/dac.3350

[41]Bian, T., Venkatesan, R., Li, C., “Design and evaluation of a new localization scheme for underwater acoustic sensor networks”, in GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference, Nov 2009, pages 1–5, available at: 10.1109/GLOCOM.2009.5425366

[42]Liu, B., Chen, H., Zhong, Z., Poor, H. V., “Asymmetrical round trip based synchronization-free localization in large-scale underwater sensor networks”, IEEE Transactions on Wireless Communications, Vol. 9, No. 11, November 2010, pages 3532–3542, available at: 10.1109/TWC.2010.090210.100146

[43]Jaffe, J. S., Franks, P. J. S., Roberts, P. L. D., Mirza, D., Schurgers, C., Kastner, R., Boch, A., “A swarm of autonomous miniature underwater robot drifters for exploring submesoscale ocean dynamics”, Nature Communications, Vol. 8, jan 2017, page 14189, available at: 10.1038/ncomms14189

[44]Roemmich, D., Owens, W. B., “The argo project: Global ocean observations for understanding and prediction of climate variability”, Oceanography, Vol. 13, No. 2, 2000, pages 45–50, available at: http://www.jstor.org/stable/43925483

[45]Babić, A., Mandić, F., Vasiljević, G., Mišković, N., “Autonomous docking and energy sharing between two types of robotic agents”, IFAC-PapersOnLine, Vol. 51, No. 29, 2018, pages 406–411, 11th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles CAMS 2018.

[46]Li, G., Svogor, I., Beltrame, G., “Self-Adaptive pattern formation with battery-powered robot swarms”, in 2017 NASA/ESA Conference on Adaptive Hardware and Systems, AHS 2017. IEEE, jul 2017, pages 253–260, available at: 10.1109/AHS.2017.8046386

[47]Arvin, F., Watson, S., Turgut, A. E., Espinosa, J., Krajník, T., Lennox, B., “Perpetual Robot Swarm: Long-Term Autonomy of Mobile Robots Using On-the-fly Inductive Charging”, Journal of Intelligent and Robotic Systems: Theory and Applications, Vol. 92, No. 3-4, dec 2018, pages 395–412, available at: 10.1007/s10846-017-0673-8

[48]Deyle, T., Reynolds, M., “Surface based wireless power transmission and bidirectional communication for autonomous robot swarms”, in Proceedings -

IEEE International Conference on Robotics and Automation. IEEE, may 2008, pages 1036–1041, available at: 10.1109/ROBOT.2008.4543341

[49]Schioler, H., Ngo, T. D., "Trophallaxis in robotic swarms - beyond energy autonomy", in 2008 10th International Conference on Control, Automation, Robotics and Vision, ICARCV 2008. IEEE, dec 2008, pages 1526–1533, available at: 10.1109/ICARCV.2008.4795751

[50]Arvin, F., Samsudin, K., Ramli, A. R., "Swarm robots long term autonomy using moveable charger", in Proceedings - 2009 International Conference on Future Computer and Communication, ICFCC 2009. IEEE, apr 2009, pages 127–130, available at: 10.1109/ICFCC.2009.48

[51]Amory, A., Tosik, T., Maehle, E., "A load balancing behavior for underwater robot swarms to increase mission time and fault tolerance", in Proceedings of the International Parallel and Distributed Processing Symposium, IPDPS. IEEE, may 2014, pages 1306–1313, available at: 10.1109/IPDPSW.2014.146

[52]Manikandan, J., Vishwanath, A., Korulla, M., "Design of a 1kW Underwater Wireless Charging Station for Underwater Data Gathering Systems", in 2018 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2018. IEEE, sep 2018, pages 211–216, available at: 10.1109/ICACCI.2018.8554936

[53]Bokc, T., Maurer, M., Farber, G., "Validation of the vehicle in the loop (VIL); a milestone for the simulation of driver assistance systems", in 2007 IEEE Intelligent vehicles symposium. IEEE, 2007, pages 612–617.

[54]Brinkmann, M., Abdelaal, M., Hahn, A., "Vessel-in-the-loop architecture for testing highly automated maritime systems", in Proceedings of the 17th Conference on Computer and IT Applications in the Maritime Industries, 2018.

[55]Tosik, T., Maehle, E., "MARS: A simulation environment for marine robotics", in 2014 Oceans - St. John's, OCEANS 2014. IEEE, sep 2015, pages 1–7, available at: 10.1109/OCEANS.2014.7003008

[56]Ridao, P., Batlle, E., Ribas, D., Carreras, M., "NEPTUNE: A HIL simulator for multiple UUVs", in Ocean '04 - MTS/IEEE Techno-Ocean '04: Bridges across the Oceans - Conference Proceedings, Vol. 1. IEEE, 2004, pages 524–531, available at: 10.1109/oceans.2004.1402970

[57]Kaliappan, V. K., Budiyono, A., Min, D., Muljowidodo, K., Nugroho, S. A., "Hardware-In-the-Loop simulation platform for the design, testing and validation of autonomous control system for unmanned underwater vehicle", Indian Journal of Marine Sciences, Vol. 41, No. 6, 2012, pages 575–580, available at: 10.21535/ProICIUS.2011.v7.380

[58]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. Kobe, Japan, 2009, page 5.

[59]Đula Nađ, Mišković, N., Mandić, F., "Navigation, guidance and control of an over-actuated marine surface vehicle", Annual Reviews in Control, Vol. 40, 2015, pages 172–181.

[60]Caccia, M., Bibuli, M., Bono, R., Bruzzone., G., "Basic navigation, guidance and control of an Unmanned Surface Vehicle", Autonomous Robots, Vol. 25, No. 4, 2008, pages 349–365.

[61]Kan, T., Mai, R., Mercier, P. P., Mi, C., "Design and Analysis of a Three-Phase

Wireless Charging System for Lightweight Autonomous Underwater Vehicles", pages 1–1, available at: 10.1109/TPEL.2017.2757015 2017.

[62]McGinnis, T., Henze, C. P., Conroy, K., "Inductive Power System for Autonomous Underwater Vehicles", in Proceedings of OCEANS'07 MTS/IEEE Conference. IEEE, sep 2007, pages 1–5, available at: 10.1109/OCEANS.2007.4449219

[63]Pinto, J., Dias, P. S., Gonçalves, R., Marques, E., Gonçalves, G., Sousa, J. B., Pereira, F. L., "NEPTUS – A Framework to Support the Mission Life Cycle", in 7th IFAC Conference on Manoeuvring and Control of Marine Craft, 2006.

[64]Barcellona, S., Brenna, M., Foiadelli, F., Longo, M., Piegari, L., "Analysis of ageing effect on li-polymer batteries", The Scientific World Journal, Vol. 2015, 2015.

[65]Pop, V., Bergveld, H. J., Op Het Veld, J. H., Regtien, P. P., Danilov, D., Notten, P. H., "Modeling battery behavior for accurate state-of-charge indication", Journal of the Electrochemical Society, Vol. 153, No. 11, nov 2006, page A2013, available at: 10.1149/1.2335951

[66]Weiss, L. E., Sanderson, A. C., Neuman, C. P., "Dynamic visual servo control of robots: An adaptive image-based approach", in Proceedings - IEEE International Conference on Robotics and Automation, Vol. 2. Institute of Electrical and Electronics Engineers, 1985, pages 662–668, available at: 10.1109/ROBOT.1985.1087296

[67]Weiss, L. E., Sanderson, A. C., Neuman, C. P., "Dynamic Sensor-Based Control of Robots with Visual Feedback", IEEE Journal on Robotics and Automation, Vol. 3, No. 5, oct 1987, pages 404–417, available at: 10.1109/JRA.1987.1087115

[68]Espiau, B., Chaumette, F., Rives, P., "A new approach to visual servoing in robotics", in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 708 LNCS P, No. 3, jun 1993, pages 106–136, available at: 10.1007/3-540-57132-9_8

[69]Sivčev, S., Rossi, M., Coleman, J., Dooly, G., Omerdić, E., Toal, D., "Fully automatic visual servoing control for work-class marine intervention ROVs", Control Engineering Practice, Vol. 74, may 2018, pages 153–167, available at: 10.1016/j.conengprac.2018.03.005

[70]Lots, J.-F., Lane, D., Trucco, E., Chaumette, F., "A 2-D Visual Servoing Technique for Underwater Vehicle Station Keeping", IFAC Proceedings Volumes, Vol. 34, No. 7, jul 2017, pages 143–148, available at: 10.1016/s1474-6670(17)35073-5

[71]Gao, J., Proctor, A. A., Shi, Y., Bradley, C., "Hierarchical Model Predictive Image-Based Visual Servoing of Underwater Vehicles with Adaptive Neural Network Dynamic Control", IEEE Transactions on Cybernetics, Vol. 46, No. 10, oct 2016, pages 2323–2334, available at: 10.1109/TCYB.2015.2475376

[72]Pisano, E. D., Zong, S., Hemminger, B. M., DeLuca, M., Johnston, R. E., Muller, K., Braeuning, M. P., Pizer, S. M., "Contrast Limited Adaptive Histogram Equalization image processing to improve the detection of simulated spiculations in dense mammograms", Journal of Digital Imaging, Vol. 11, No. 4, 1998, page 193, available at: 10.1007/BF03178082

[73]Reza, A. M., "Realization of the Contrast Limited Adaptive Histogram Equalization (CLAHE) for Real-Time Image Enhancement", Journal of VLSI signal processing systems for signal, image and video technology, Vol. 38, No. 1, 2004, pages 35–44, available at: 10.1023/B:VLSI.0000028532.53893.82

[74]Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S.,

Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mane, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viegas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems", mar 2016.

[75]Tzutalin, "Labelimg", Free Software: MIT License, available at: https://github.com/tzutalin/labelImg 2015.

[76]Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L. C., "MobileNetV2: Inverted Residuals and Linear Bottlenecks", in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, jan 2018, pages 4510–4520, available at: 10.1109/CVPR.2018.00474

[77]Girshick, R., "Fast R-CNN", in Proceedings of the IEEE International Conference on Computer Vision, Vol. 2015 Inter, apr 2015, pages 1440–1448, available at: 10.1109/ICCV.2015.169

[78]Redmon, J., Farhadi, A., "YOLO v.3: An Incremental Improvement", Tech report, 2018, pages 1–6.

[79]Agrawal, P., Girshick, R., Malik, J., "Analyzing the performance of multilayer neural networks for object recognition", in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 8695 LNCS, No. PART 7. Springer, Cham, 2014, pages 329–344, available at: 10.1007/978-3-319-10584-0_22

[80]Rojas, R., "The Kalman filter", The Mathematical Intelligencer, Vol. 1, No. 2, 1978, pages 90–92, available at: 10.1007/BF03023070

[81]Ting Goh, S., Reza Zekavat, S. A., Abdelkhalik, O., "An Introduction to Kalman Filtering Implementation for Localization and Tracking Applications", in Handbook of Position Location. Wiley, jan 2019, pages 143–195, available at: 10.1002/9781119434610.ch5

[82]Pinto, J., Dias, P. S., Gonçalves, R., Marques, E., Gonçalves, G., Sousa, J. B., Pereira, F. L., "NEPTUS – A Framework to Support the Mission Life Cycle", in 7th IFAC Conference on Manoeuvring and Control of Marine Craft, 2006.

[83]Pisinger, D., Ropke, S., "A general heuristic for vehicle routing problems", Computers and Operations Research, Vol. 34, No. 8, 2007, pages 2403–2435, available at: 10.1016/j.cor.2005.09.012

[84]Berhan, E., Krömer, P., Kitaw, D., Abraham, A., Snášel, V., "Solving Stochastic Vehicle Routing Problem with Real Simultaneous Pickup and Delivery Using Differential Evolution", Advances in Intelligent Systems and Computing, Vol. 237, 2014, pages 187–200, available at: 10.1007/978-3-319-01781-5

[85]Tuzun, D., Magnet, M. A., Burke, L. I., "Selection of vehicle routing heuristic using neural networks", pages 211–221, 1997.

[86]Gutierrez-Rodríguez, A. E., Conant-Pablos, S. E., Ortiz-Bayliss, J. C., Terashima-Marín, H., "Selecting meta-heuristics for solving vehicle routing problems with time windows via meta-learning", Expert Systems with Applications, Vol. 118, No. October, 2019, pages 470–481, available at: 10.1016/j.eswa.2018.10.036

[87]Ghesmoune, M., Lebbah, M., Azzag, H., "State-of-the-art on clustering data streams", Big Data Analytics, Vol. 1, No. 1, dec 2016, page 13, available at:

10.1186/s41044-016-0011-3

[88]Wong, K.-C., "A Short Survey on Data Clustering Algorithms", in 2015 Second International Conference on Soft Computing and Machine Intelligence (ISCMI), 2015, pages 64–68, available at: 10.1109/ISCMI.2015.10

[89]Jain, A. K., "Data clustering: 50 years beyond K-means", Pattern Recognition Letters, Vol. 31, 2009, pages 651–666, available at: 10.1016/j.patrec.2009.09.011

[90]Bair, E., "Semi-supervised clustering methods", 2013, pages 1–28, available at: 10.1002/wics.1270

[91]Bradley, P. S., Bennett, K. P., Demiriz, A., "Constrained k-means clustering", Tech. Rep., 2000, available at: 10.1016/S0025-7753(14)70064-8

[92]Zhi, W., Wang, X., Qian, B., Butler, P., Ramakrishnan, N., Davidson, I., "Clustering with Complex Constraints - Algorithms and Applications", Proceedings of the 27th AAAI Conference on Artificial Intelligence, July 14-18, 2013, Washington, USA, 2013, pages 1056–1062.

[93]Basu, S., Banerjee, A., Mooney, R. J., "Active Semi-Supervision for Pairwise Constrained Clustering", 2004, pages 333–344, available at: http://www.cs.utexas.edu/{~}ml/papers/semi-sdm-04.pdf

[94]Hsu, Y.-C., Kira, Z., "Neural network-based clustering using pairwise constraints", 2015, pages 1–12, available at: http://arxiv.org/abs/1511.06321

[95]Sandeep, D. N., Kumar, V., "Review on Clustering, Coverage and Connectivity in Underwater Wireless Sensor Networks: A Communication Techniques Perspective", pages 11 176–11 199, available at: 10.1109/ACCESS.2017.2713640 2017.

[96]Storn, R., Price, K., "Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces", Journal of Global Optimization, Vol. 11, No. 4, 1997, pages 341–359, available at: 10.1023/A:1008202821328

[97]Nearchou, A. C., Omirou, S. L., "Differential evolution for sequencing and scheduling optimization", Journal of Heuristics, Vol. 12, No. 6, dec 2006, pages 395–411, available at: 10.1007/10732-006-3750-x

[98]Onwubolu, G. C., Davendra, D., Differential evoluiton: A handbook for global permutation-ased combinatorial optimization, 2009, available at: 10.1007/978-3-540-92151-6

[99]Kromer, P., Abraham, A., Snasel, V., Berhan, E., Kitaw, D., "On the differential evolution for vehicle routing problem", 2013 International Conference on Soft Computing and Pattern Recognition (SoCPaR), 2013, pages 384–389, available at: 10.1109/SOCPAR.2013.7054163

[100]Mingyong, L., Erbao, C., "An improved differential evolution algorithm for vehicle routing problem with simultaneous pickups and deliveries and time windows", Engineering Applications of Artificial Intelligence, Vol. 23, No. 2, 2010, pages 188–195, available at: 10.1016/j.engappai.2009.09.001

[101]Mezura-Montes, E., Velazquez-Reyes, J., Coello Coello, C., "Modified Differential Evolution for Constrained Optimization", IEEE International Conference on Evolutionary Computation, No. 70771037, 2006, pages 25–32, available at: 10.1109/CEC.2006.1688286

[102]Müller, K., Vignaux, T., Lünsdorf, O., Scherfke, S., "SimPy Documentation", 2018, page 373, available at: https://simpy.readthedocs.io/en/latest/

[103]Burke, E. K., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., Woodward, J. R., "A Classification of Hyper-heuristics Approaches", Handbook of Metaheuristics,

Vol. 57, 2010, pages 449–468, available at: doi:10.1007/978-1-4419-1665-5_15

[104]Cowling, P., Kendall, G., Soubeiga, E., "A Hyperheuristic Approach to Scheduling a Sales Summit". Springer, Berlin, Heidelberg, 2001, pages 176–190, available at: 10.1007/3-540-44629-X_11

[105]Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P., Schulenburg, S., "Hyper-Heuristics: An Emerging Direction in Modern Search Technology", in Handbook of Metaheuristics. Boston: Kluwer Academic Publishers, 2003, pages 457–474, available at: 10.1007/0-306-48056-5_16

[106]Chakhlevitch, K., Cowling, P., "Hyperheuristics: Recent developments", Berlin, Heidelberg, pages 3–29, available at: 10.1007/978-3-540-79438-7_1 2008.

[107]Burke, E., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Qu, R., "Hyper-heuristics: a survey of the state of the art", Journal of the Operational Research Society, Vol. 64, No. 12, dec 2013, pages 1695–1724, available at: 10.1057/jors.2013.71

[108]Pillay, N., Qu, R., "Assessing Hyper-Heuristic Performance", Journal of the Operational Research Society, Vol. 72, No. 11, nov 2021, pages 2503–2516, available at: 10.1080/01605682.2020.1796538

[109]Ryser-Welch, P., Miller, J., "A review of hyper-heuristic frameworks", 04 2014.

[110]Drake, J. H., Kheiri, A., Özcan, E., Burke, E. K., "Recent Advances in Selection Hyper-heuristics", European Journal of Operational Research, Vol. 285, No. 2, aug 2019, pages 405–428, available at: 10.1016/J.EJOR.2019.07.073

[111]Zhang, C., Zhao, Y., Leng, L., "A hyper-heuristic algorithm for time-dependent green location routing problem with time windows", IEEE Access, Vol. 8, 2020, pages 83 092–83 104, available at: 10.1109/ACCESS.2020.2991411

[112]Burke, E., Curtois, T., Hyde, M., Kendall, G., Ochoa, G., Petrovic, S., Vazquez-Rodriguez, J., "HyFlex: A flexible framework for the design and analysis of hyper-heuristics", in Multidisciplinary International Scheduling Conference (MISTA 2009), 2009, pages 790–797, available at: https://pdfs.semanticscholar.org/c456/64487c419c52fb72f77382f1c912478975a6.pdfhttp://www.mistaconference.org/2009/abstracts/790-797-112-A.pdf

[113]Asta, S., Ozcan, E., "An apprenticeship learning hyper-heuristic for vehicle routing in HyFlex", in IEEE SSCI 2014 - 2014 IEEE Symposium Series on Computational Intelligence - EALS 2014: 2014 IEEE Symposium on Evolving and Autonomous Learning Systems, Proceedings. IEEE, dec 2014, pages 65–72, available at: 10.1109/EALS.2014.7009505

[114]Pierreval, H., "Neural Network to Select Dynamic Scheduling Heuristics", Journal of Decision Systems, Vol. 2, No. 2, jan 1993, pages 173–190, available at: 10.1080/12460125.1993.10511572

[115]Ortiz-Bayliss, J. C., Terashima-Mar??n, H., Conant-Pablos, S. E., "Neural networks to guide the selection of heuristics within constraint satisfaction problems", in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 6718 LNCS, 2011, pages 250–259, available at: 10.1007/978-3-642-21587-2_27

[116]Tyasnurita, R., Ozcan, E., John, R., "Learning heuristic selection using a Time Delay Neural Network for Open Vehicle Routing", in 2017 IEEE Congress on Evolutionary Computation, CEC 2017 - Proceedings. IEEE, jun 2017, pages 1474–1481, available at: 10.1109/CEC.2017.7969477

[117]Maashi, M. S., "Multi-Objective Hyper-Heuristics", in Heuristics and Hyper-Heuristics - Principles and Applications, 2017, available at: 10.5772/intechopen.69222

[118]Asta, S., Ozcan, E., Parkes, A. J., Ima Etaner-Uyar, A., "Generalizing Hyper-heuristics via Apprenticeship Learning", 2013, available at: http://cs.nott.ac.uk/http://web.itu.edu.tr/http://cs.nott.ac.uk/{%}0Ahttp://web.itu.edu.tr/

[119]Burke, E. K., Hyde, M. R., Kendall, G., Ochoa, G., Özcan, E., Woodward, J. R., "A classification of hyper-heuristic approaches: Revisited", in International Series in Operations Research and Management Science. Springer International Publishing AG, 2019, Vol. 272, pages 453–477, available at: 10.1007/978-3-319-91086-4_14

[120]Sánchez-Díaz, X., Ortiz-Bayliss, J. C., Amaya, I., Cruz-Duarte, J. M., Conant-Pablos, S. E., Terashima-Marin, H., "A feature-independent hyper-heuristic approach for solving the knapsack problem", Applied Sciences (Switzerland), Vol. 11, No. 21, 2021, page 10209, available at: 10.3390/app112110209

[121]Kheiri, A., Mısır, M., Özcan, E., "Ensemble move acceptance in selection hyper-heuristics", in Communications in Computer and Information Science, Vol. 659, 2016, pages 21–29, available at: 10.1007/978-3-319-47217-1_3

[122]Drake, J. H., Özcan, E., Burke, E. K., "An improved choice function heuristic selection for cross domain heuristic search", in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 7492 LNCS, No. PART 2, 2012, pages 307–316, available at: 10.1007/978-3-642-32964-7_31

[123]Demeester, P., Bilgin, B., De Causmaecker, P., Van Den Berghe, G., "A hyperheuristic approach to examination timetabling problems: Benchmarks and a new problem from practice", Journal of Scheduling, Vol. 15, No. 1, 2012, pages 83–103, available at: 10.1007/s10951-011-0258-5

[124]de Carvalho, V. R., Özcan, E., Sichman, J. S., "Comparative analysis of selection hyper-heuristics for real-world multi-objective optimization problems", Applied Sciences (Switzerland), Vol. 11, No. 19, 2021, available at: 10.3390/app11199153

[125]de Santiago Júnior, V. A., Özcan, E., de Carvalho, V. R., "Hyper-Heuristics based on Reinforcement Learning, Balanced Heuristic Selection and Group Decision Acceptance", Applied Soft Computing Journal, Vol. 97, 2020, available at: 10.1016/j.asoc.2020.106760

[126]Alanazi, F., Lehre, K. P., "Limits to learning in reinforcement learning hyper-heuristics", Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 9595, 2016, pages 170–185, available at: 10.1007/978-3-319-30698-8_12

[127]Qin, W., Zhuang, Z., Huang, Z., Huang, H., "A novel reinforcement learning-based hyper-heuristic for heterogeneous vehicle routing problem", Computers and Industrial Engineering, Vol. 156, No. March, jun 2021, page 107252, available at: 10.1016/j.cie.2021.107252

[128]Paolo, G., "Learning in Sparse Rewards settings through Quality-Diversity algorithms", PhD thesis, Sorbonne Université, 2022, available at: http://arxiv.org/abs/2203.01027

[129]Kalender, M., Kheiri, A., Özcan, E., Burke, E. K., "A greedy gradient-simulated annealing selection hyper-heuristic", Soft Computing, Vol. 17, No. 12, 2013, pages 2279–2292, available at: 10.1007/s00500-013-1096-5

Acronyms

API	Application Programming Interface	24
CMA	Cumulative Moving Average	84
DE	Differential Evolution	59
GPS	Global Positioning System	16
HIL	Hardware–In–the–Loop	15
IMU	Inertial Measurement Unit	16
IR	Infra-Red	17
JSON	JavaScript Object Notation	77
PCB	printed circuit board	21
RGB	Red-Green-Blue	17
ROS	Robot Operating System	22
ROV	Remotely Operated underwater Vehicle	29
RTK	Real-Time Kinematic	16
SSE	Sum of Squared Error	68
UASN	Underwater Acoustic Sensor Network	12
UDP	User Datagram Protocol	77
VIL	Vehicle–In–the–Loop	15

List of Figures

List of Tables

Biography

Anja Babić was born on June 22, 1991 in Zagreb, Croatia. In 2014 she graduated from the University of Zagreb, Faculty of Electrical Engineering and Computing (FER) with a master thesis entitled "Autonomous Task Execution within NAO Robot Scouting Mission Framework". Since 2015 she has been a researcher and PhD student at FER and a member of the Laboratory for Underwater Systems and Technologies (LABUST). In 2017 she was a visiting researcher at the Consiglio Nazionale delle Ricerche in Genoa, Italy. She is a senior researcher in the project "Multifunctional Smart Buoys" and was previously involved in EU H2020 projects subCULTron - Submarine Cultures Perform Long-term Robotic Exploration of Unconventional Environmental Niches and H2020 EXCELLABUST. She participated in developing diver-focused sensing, data processing, and underwater communication as part of the FP7 project CADDY – Cognitive Autonomous Diving Buddy, as well as implementing tasks for a robot-assisted autism spectrum disorder diagnostic protocol using the humanoid robot NAO. Her research interests include evolutionary and bio-inspired robotics, emergent behaviour, task allocation and scheduling, and communication between both heterogeneous agents and members of a swarm, as applied to marine robotic platforms. Since 2019 she has been the Chair of the IEEE Oceanic Engineering Society Student Branch Chapter of the University of Zagreb.

List of publications (Anja Babić)

Journal papers

[1]Babić, A., Mandić, F., Mišković, N., "Development of visual servoing-based autonomous docking capabilities in a heterogeneous swarm of marine robots", Applied Sciences (Switzerland), Vol. 10, No. 20, 2020, pages 1–26, available at: 10.3390/app10207124

[2]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 (Switzerland), Vol. 20, No. 16, aug 2020, pages 1–23, available at: 10.3390/s20164615

[3]Babić, A., Vasiljević, G., Mišković, N., "Vehicle-in-the-Loop Framework for Testing Long-Term Autonomy in a Heterogeneous Marine Robot Swarm", IEEE Robotics and Automation Letters, Vol. 5, No. 3, jul 2020, pages 4439–4446, available at: 10.1109/LRA.2020.3000426

[4]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, apr 2019, page 1388, available at: 10.3390/app9071388

[5]Gomez Chavez, A., Ranieri, A., Chiarella, D., Zereik, E., Babić, A., Birk, A., "CADDY Underwater Stereo-Vision Dataset for Human–Robot Interaction (HRI) in the Context of Diver Activities", Journal of Marine Science and Engineering, Vol. 7, No. 1, jan 2019, page 16, available at: 10.3390/jmse7010016

Conference papers

[1]Babić, A., Ferreira, F., Kapetanović, N., Mišković, N., Bibuli, M., Corrado, M., Ferretti, R., Odetti, A., Caccia, M., Aracri, S., De Pascalis, F., "Cooperative marine litter detection and environmental monitoring using heterogeneous robotic agents", in OCEANS 2023 Limerick, 2023, accepted for publication.

[2]Ferreira, F., Babić, A., Oreč, M., Mišković, N., Motta, C., Ferretti, R., Odetti, A., Aracri, S., Bruzzone, G., Caccia, M., Braga, F., Manfè, G., Lorenzetti, G., Scarpa, G., De Pascalis, F., "Heterogeneous marine robotic system for environmental monitoring missions", in 2023 IEEE Underwater Technology (UT), 2023, accepted for publication.

[3]Babić, A., Oreč, M., Mišković, N., "Developing the concept of multifunctional smart buoys", OCEANS 2021: San Diego – Porto, sep 2021, pages 1–6, available at: 10.23919/OCEANS44145.2021.9705916

[4]Borković, G., Fabijanić, M., Magdalenić, M., Malobabić, A., Vuković, J., Zielinski, I., Kapetanović, N., Kvasić, I., Babić, A., Mišković, N., "Underwater ROV Software for Fish Cage Inspection", in 2021 44th International Convention on Information, Communication and Electronic Technology, MIPRO 2021 - Proceedings. IEEE, sep 2021, pages 1747–1752, available at: 10.23919/MIPRO52101.2021.9596823

[5]Babić, A., Mandić, F., Vasiljević, G., Mišković, N., "Autonomous docking and energy sharing between two types of robotic agents", IFAC-PapersOnLine, Vol. 51, No. 29, 2018, pages 406 - 411, 11th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles CAMS 2018.

[6]Babić, A., Mišković, N., Vukić, Z., "Heuristics pool for hyper-heuristic selection during task allocation in a heterogeneous swarm of marine robots", IFAC-PapersOnLine, Vol. 51, No. 29, jan 2018, pages 412–417, available at: 10.1016/j.ifacol.2018.09.452

[7]Thenius, R., Moser, D., Varughese, J. C., Kernbach, S., Kuksin, I., Kernbach, O., Kuksina, E., Mišković, N., Bogdan, S., Petrović, T., Babić, A., Boyer, F., Lebastard, V., Bazeille, S., Ferrari, G. W., Donati, E., Pelliccia, R., Romano, D., van Vuuren, G. J., Stefanini, C., Morgantin, M., Campo, A., Schmickl, T., "subCULTron - Cultural development as a tool in underwater robotics", in Communications in Computer and Information Science, Vol. 732, 2018, pages 27–41, available at: 10.1007/978-3-319-90418-4_3

[8]Chavez, A. G., Mueller, C. A., Birk, A., Babić, A., Mišković, N., "Stereo-vision based diver pose estimation using LSTM recurrent neural networks for AUV navigation guidance", in OCEANS 2017 - Aberdeen, Vol. 2017-Octob, 2017, pages 1–7, available at: 10.1109/OCEANSE.2017.8085020

[9]Babić, A., Jagodin, N., Kovačić, Z., "Autonomous task execution within NAO robot scouting mission framework", in 2017 European Conference on Mobile Robots (ECMR). Paris, France: IEEE, sep 2017, pages 1–7, available at: 10.1109/ECMR.2017.8098705

[10]Babić, A., Lončar, I., Mišković, N., Vukić, Z., "Energy-efficient environmentally adaptive consensus-based formation control with collision avoidance for multi-vehicle systems", IFAC-PapersOnLine, Vol. 49, No. 23, jan 2016, pages 361–366, available at: 10.1016/j.ifacol.2016.10.431

[11]Mišković, N., Vukić, Z., Bogdan, S., Babić, A., "subSULTron: Submarine cultures perform long-term robotic exploration of unconventional environmental niches", in

Proceedings of the 6th Conference on Marine Technology in memory of Academician Zlatko Winkler. Rijeka, Croatia: 6th Conference on Marine Technology in memory of Academician Zlatko Winkler, 2015, available at: http://bib.irb.hr/prikazi-rad?rad=788247

[12]Petrić, F., Hrvatinić, K., Babić, A., Malovan, L., Miklić, D., Kovačić, Z., Cepanec, M., Stošić, J., Šimleša, S., "Four tasks of a robot-assisted autism spectrum disorder diagnostic protocol: First clinical tests", in Proceedings of the 4th IEEE Global Humanitarian Technology Conference, GHTC 2014. IEEE, oct 2014, pages 510–517, available at: 10.1109/GHTC.2014.6970331

Životopis

Anja Babić rođena je 22. lipnja 1991. u Zagrebu. 2014. godine diplomirala je na Sveučilištu u Zagrebu, Fakultetu elektrotehnike i računarstva (FER) s diplomskim radom pod naslovom „Autonomno izvršavanje zadataka u okviru izviđačke misije NAO robota“. Od 2015. istraživačica je i doktorandica na FER-u te članica Laboratorija za podvodne sustave i tehnologije (LAPOST). Godine 2017. bila je gostujući istraživač na Consiglio Nazionale delle Ricerche u Genovi, Italija. Viša je istraživačica na projektu “Multifunkcionalne pametne bove”, a prethodno je bila uključena u EU H2020 projekte subCULTron - Submarine Cultures Perform Long-term Robotic Exploration of Unconventional Environmental Niches i H2020 EXCELLABUST. Sudjelovala je u razvoju senzorskih sustava, obradi podataka i ostvarenju podvodne komunikacije usmjerenih na ronioce u sklopu FP7 projekta CADDY – Cognitive Autonomous Diving Buddy, kao i implementaciji zadataka za robotski potpomognuti dijagnostički protokol poremećaja iz autističnog spektra pomoću humanoidnog robota NAO. Njezini istraživački interesi uključuju evolucijsku i bio-inspiriranu robotiku, planiranje misija, dodjeljivanje i raspodjelu zadataka te komunikaciju između heterogenih agenata kao i članova roja, uz naglasak na primjenu na morske robotske platforme. Od 2019. predsjednica je studentskog odjela IEEE Oceanic Engineering Society Sveučilišta u Zagrebu.